MASTER IN COMPUTER SCIENCE

Automatic analysis of interferograms

Pirenne, Benoît

Award date:
1986

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Namur, Belgium.

European Southern Observatory
Garching bei München, Germany

AUTOMATIC ANALYSIS OF

INTERFEROGRAMS

Mémoire présenté par
Benoît PIRENNE
en vue de l'obtention du titre de
Licencié et Maître en Informatique.
Promoteur : M^{me} M. Noirhomme.
Année académique 1985-1986.

Acknowledgements

I would like to thank the ESO (European Southern Observatory) for the studentship it offered to me and without which this document wouldn't be born.

My highest gratitude goes to M^r J. Daniel PONZ who supported this work for its whole duration, bringing helpfull aids as well as advices and encouragements.
In the same way, I would like to thank M^r Hans DEKKER, the "customer" of this work. My hope is that he will find it useful for its needs. M^r Fionn MURTAGH and M^r Fritz MERCKLE can also receive my sincere gratitude for helpfull discussions.

M^r Klaus BANSE, who is at the source of MIDAS also receives my admiration since his image processing package has revealed to be a powerful "development bed".

Special thanks also to M^r Ph. CRANE who managed to provide me ways of subsistance as well as a link to ESO's personnel.

And lastly, to all ESO's staff members for their kindness and constant good mood.

At the University side, my best gratitude goes to M^{rs} NOIRHOMME, who remotely supported this work.

Last but not least, I have to thank the VAX editor and LaTeX text processor that gave me the ability of presenting a well formed document. I found a helpful aid in [[tpu/editor]] and [[latex]].

Contents

1 INTRODUCTION **3**

1.1 Some Theory ... 4

1.2 Interferometry and optical metrology . . . 6

1.3 Literature review . . . 9

1.3.1 Historical considerations . . . 9

1.3.2 Modern approaches . . . 10

2 OPPORTUNITY ANALYSIS **12**

2.1 ESO actual requests in automated interferometry . . . 12

2.2 Design of a global solution for both recording and analysis of interferograms . . . 13

2.2.1 Introduction : which difficulties will we encounter ? . 13

2.2.2 An idea 13

2.2.3 Draw of a solution. . . . 14

2.2.4 Conclusions . . . 14

2.3 One possible useful sub-system . . . 15

3 FUNCTIONNAL ANALYSIS **17**

3.1 Information structuration . . . 17

3.2 Process structuration . . . 18

3.3 Process dynamic . . . 21

3.4 The processes static . . . 22

3.5 Conclusions . . . 27

4 IMPLEMENTATION ANALYSIS **28**

4.1 introduction . . . 28

4.2 Full description of the first step : Phase 3 . . . 29

4.2.1 Function a: [—> search for the peaks in the interferogram] . . . 29

4.2.2 Function b: [—> thinning of the fringe structure] . . 44
4.2.3 General conclusions about this phase. 45
4.3 Full description of the second step Phase 4 45
4.3.1 introduction 45
4.3.2 Phase 4, Function a: [—> table with the positions] . 47
4.3.3 Phase 4, Function b: [—> MST] 49
4.3.4 Phase 4, Function c: [—> Split] 52
4.3.5 Phase 4, Function d: [—> Merge] 55
4.3.6 Phase 4, function e: [—> Interactive tree editing facility] 61
4.3.7 Phase 4, function f: [—> Fringe ordering] . . 65
4.3.8 General conclusions about this phase. 68
4.4 full description of the third step: Phase 5. 68
4.4.1 Introduction. 68
4.4.2 Phase 5, function a: [—> Compute aberrations] . . . 68

5 RESULTS and CONCLUSIONS **71**
5.1 Implementation 71
5.2 Advantages among other methods. 72
5.3 Future developments 73
5.4 Other fields of application 74
5.5 Results 75

A Description of the System Used. **78**
A.1 Hardware. 78
A.2 Software. 80
A.2.1 Operating System. 80
A.2.2 MIDAS – the ESO's Image Data Analysis system. . . 80

B Data Structures Used **84**

C Performance Tests (time consummations) **89**
C.1 Why testing the performance ? 89
C.2 Some results. 90
C.3 Conclusions. 90

D User's manual **91**
D.1 Introduction 92
D.2 Analysis Method 93
D.2.1 Global structure - input data - output data 93

D.2.2 Thin structure - the related algorithms. 93
D.3 Operation 104
D.3.1 Introduction 104
D.3.2 Shematic execution diagram 104
D.3.3 The level 2 ANALYSE/INTERFEROGRAM command 106
D.3.4 Operation with the level 1 command 108
D.3.5 How to store an interferogram image on disk ? 108
D.4 Implementation 109
D.4.1 Images 109
D.4.2 Tables 110
D.4.3 Other files 110
D.4.4 Commands 110
D.4.5 keywords 111

E Glossary of terms **112**

F Listings **120**

Presentation of the ESO

ESO is the acronym for **E**uropean **S**outhern **O**bservatory. As indicated by its name, ESO is a European organization, financed by Belgium, Germany, Italy, Denmark, Switzerland, the Nederlands, France and Sweden. ESO is essentially a high level tool for astronomical research in the Southern hemisphere. This institution has two different locations: the first one in Garching, near Munich (F.R.G.) and the second one in La Silla in Chile. Most of the instruments and astronomical devices are designed, built and tested in Garching while the observations take place in the Andes mountains in Chile. Moreover, many of the results of the observations are brought back from Chile to Garching, where the astronomers can find the best facilities to reduce their data.

For more informations, consult the ESO pamphlet, available on simple demand, in any language at ESO:

Karl-Schwarzchild-Straße, 2
D-8046 Garching bei München
GERMANY

Scope of this document

The aim of this book is to present and describe as clearly as possible the different steps that led to the realization of a useful and efficient computer software for the analysis of interferograms.
In a first chapter, the reader will enter the world of modern optical measurements and learn about interferometry and its applications for the quality control of mirrors and lenses. This part will also present different ways used on the present day to analyse the so called interferograms, underlining their disadvantages and facilities.

The three following chapters explains the new method that we have built for the **automatic analysis of interferograms** and will justify this title. Each of these chapters represents one main step in the process of modern software development. The method used for the definition of this package has been described in [[bodart83]]. For a description of the different functions of the processes, some idea from [[van lamsweerde84]] found here a place to express while, for the explanations of the algorithms, both "pseudo-code" and "programatic tree" methods could successfully be applied. All this resulted in a very modular design, from the highest levels to the smallest sub-routine.

The fith chapter, after some considerations about the logical implementation, presents also some possible improvements and describes other fields of applications for this package. Then, some results are presented, showing the different steps of the analysis process on gathered images.

The appendices will describe the system used (from both hard- and software point of view); the data structures manipulated by the application; the performance of the whole process; a users' manual; a glossary of the technical and specific terms used and lastly the program codes. The bibliography takes place at the end of this book.

Chapter 1

INTRODUCTION

"Une introduction doit toujours à la fois ouvrir la lecture de ce qui va suivre, tout en en donnant le tempo."
J. Berleur

This first chapter, as indicated by its name, aims at giving you the basic knowledge and general information necessary to be able to understand the rest of this work and the motivation of its existence.
In a first section, we will recall you some basics about the physics of optics, necessary to introduce the interference phenomenon. The reader who would skip this first section will not be much disadvantaged with respect to the other, more brave, readers since the other chapters will not refer to this theory anymore[1].

The next section, devoted to the applications of interferometry, will emphasize its usefulness for optical metrology. This point is more important, since it is the main field of application of this work. It explains how one can use interferometry techniques for controlling optical quality of mirrors and system lenses. In the last section a short literature review is presented. It will explain how different methods now available do actually proceed to extract from these interferograms a useful and correct interpretation of the quality of the optics tested.

[1] As a matter of fact, the main problems will consist of pattern recognition problems, as explained later on.

1.1 Some Theory ...

Everybody should be aware of the main characteristics of light: its wavy behaviour and its corpuscular character. We won't retain this last feature but devote our time to a particular phenomenon generated by the first property of light: the wavy movement. It involves at least one main consequence: the *interference phenomenon*. It does happen when two or more oscillating movements exist together in time and space. To show the consequences of such a property, let us write the equation of a sine wave:

$$\Psi(x,t) = A\sin(kx - \omega t + \epsilon)$$

were A is the wave amplitude, $k = \frac{2\pi}{\lambda}$ with λ = wavelength, ω is the angular frequency expressed in radians per second, introduced here because a sine wave is periodic and because we have:

$$\Psi(x,t) = \Psi(x \mp \lambda, t)$$

ϵ will be here the expression of the phase (translation of the sine function by giving a displacement ϵ at time $t = 0$).

If we now dispose over two wavefront with the same wavelength emitted by remote sources and crossing at point p (see fig. 1.1), we can write:

$$\Psi_1(x,t) = A_1\sin(kx_1 - \omega t + \epsilon_1)$$

$$\Psi_2(x,t) = A_2\sin(kx_2 - \omega t + \epsilon_2)$$

Let us now call:

$$\delta = kx_1 - kx_2 + \epsilon_1 - \epsilon_2 = \frac{2\pi}{\lambda}(x_1 - x_2)$$

We can then write the resulting amplitude in p like this:

$$\Psi = \sqrt{\Psi_1^2 + \Psi_2^2 + 2\Psi_1\Psi_2\cos\delta}$$

It is then clear that :
$\Psi = \Psi_1 - \Psi_2$ if $\cos\delta = -1$ i.e. $\delta = 2(n+1)\pi \quad n \in \mathcal{Z}$

$\Psi = \Psi_1 + \Psi_2$ if $\cos\delta = +1$ i.e. $\delta = 2n\pi \quad n \in Z$

$$\Psi_1 - \Psi_2 < \Psi < \Psi_1 + \Psi_2$$

otherwise.
In the former case, we have a maximal strenghtening of both vibrating movements that is, a *constructive interference* and, in the latter case a maximal attenuation or *destructive interference*. So, we have:

$$\delta = \begin{cases} 2n\pi & constructive\ interference \\ (2n+1)\pi & destructive\ interference \end{cases}$$

that we can write

$$\frac{2\pi}{\lambda}(x_1 - x_2) = \begin{cases} 2n\pi \\ (2n+1)\pi \end{cases}$$

or

$$x_1 - x_2 = \begin{cases} n\lambda \\ (2n+1)\frac{\lambda}{2} \end{cases}$$

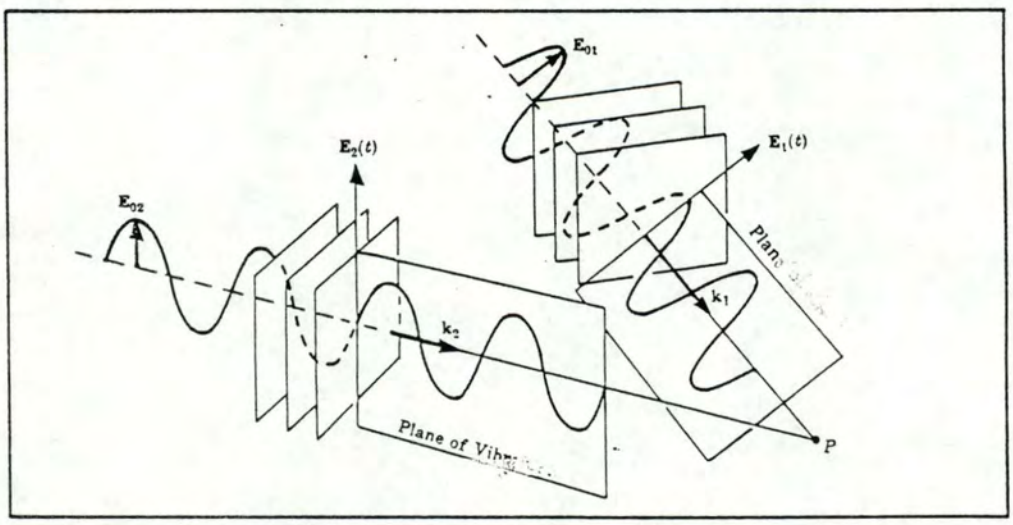
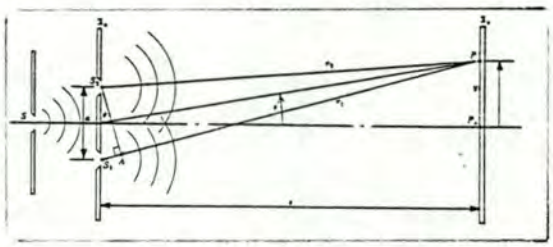


fig 1.1 : two wavefront with the same wavelenth crossing at point p

With such results, we can now explain how one can construct a simple interferometer, called *Young interferometer.*
As shown on fig. 1.2 hereunder, it is composed of a point source emiting light divided into two different point sources that can now interfere. The results of such an experiment appear on a screen set on the right of the device. The last figure (1.3) shows the so called fringes, alternatively black and white, alternatively destructive and constructive interferences.



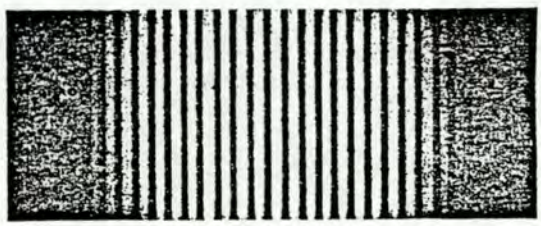


Fig. 1.2 Young interferometer
Fig. 1.3 succession of constructive and destructive interferences.

This section was a compilation of the chapters related to interferometry in [[alonso67]] and in [[hecht75]]. The reader who wishes to know more about this subject will find interesting and complete developments in these books, as well as in any serious book dedicated to physics.

1.2 Interferometry and optical metrology

Up to now, we have presented the interferometry phenomenon but we don't yet know how it can be useful. To discover it, let us cite [[wyant82]]:

" The interferometer shown in fig. 1.4 is commonly used for measuring the variation between a test surface and a reference surface. The test surface can be a flat, spherical, or even an aspheric surface. As shown in the figure, the surface under test is placed in contact, or at least near contact, with a reference surface of known high quality. Generally a small air wedge exists between the two surfaces giving some 8 to 15 fringes having a sinusoidal intensity profile as illustrated in figure 1.3. "

" If the two surfaces match, straight equally spaced fringes result. Surface height variations between the two surfaces cause the fringes to deviate from straightness and/or equal separation, where one fringe deviation from straightness correspond to an optical path difference equal to the wavelength of light used in the test; i.e. a height variation of one-half wavelength. The wavelength of a helium source, witch is often used in such an interferometer, is 587.56 nanometers, hence one fringe correspond to a height variation of approximately 0.3 micrometers. "

" Fig. 1.5 illustrates interference fringes obtained using this kind of interferometer. The fringes have an average spacing of S and a deviation in spacing of $\Delta(x, y)$ where the value of $\Delta(x, y)$ varies over the interferogram. For a point where the deviation is Δ, the surface height error is given by

$$\text{Surface height error} = \frac{\lambda}{2}\frac{\Delta}{S}$$

With the naked eye, values of $\frac{\Delta}{S}$ of $1/10$ to $1/20$ can be determined. "

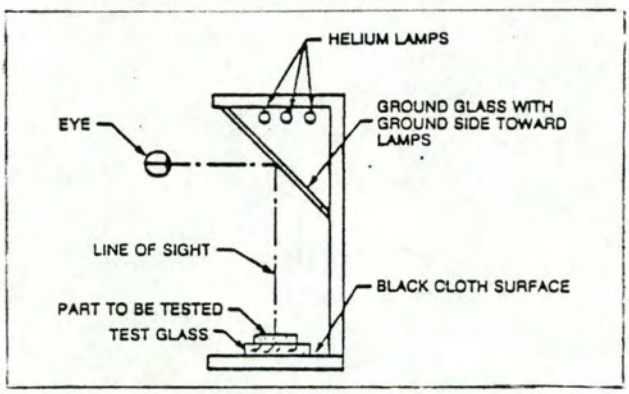


fig. 1.4 A simple kind of interferometer.

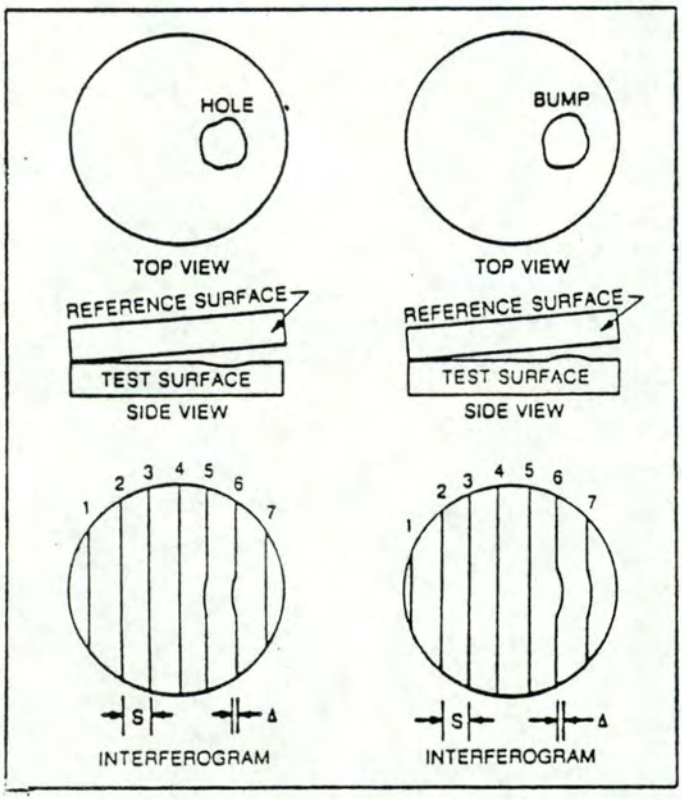


fig. 1.5 effects of a height variation of the surface to test in the shape of the fringes.

So we have seen that interferometry can be used to accurately determine the "flatness" of a mirror or of a lens: variations in the fringe shape testify to imperfections on the optic. Citing the Wyant's article, we have shown one type of interferometer. But there are many others, giving the same type of results, using the same principle. The *Twyman-Green* interferometer (fig. 1.6) is the type currently used at ESO. It allows both tests of spherical mirrors and lenses.

We can also remark that now modern interferometer use laser light since

the laser provides coherent and monochromatic light beam.

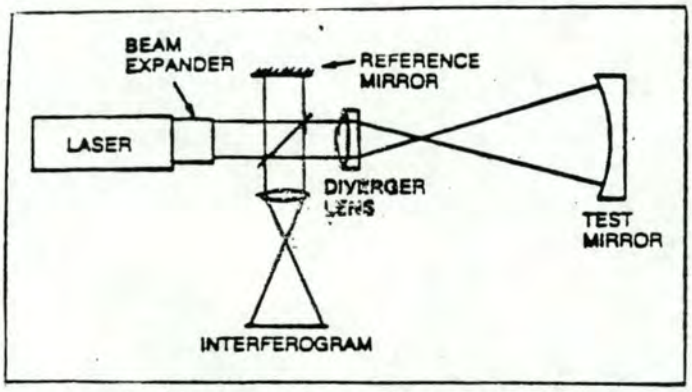


fig 1.6 A typical Twyman-Green interferometer.

1.3 Literature review

1.3.1 Historical considerations

The first, the fastest and the easiest way to judge of the quality of an optical component is to use the naked human eye! As a matter of fact, it can determine the type of aberration presented as well as its amplitude to an accuracy of $\lambda/10$. But that's all. To reach better values or to find out the repartition of many mixed aberrations, a careful analysis is then mandatory. This can be done manually or more or less automatically. Let us cite [[becker85]] who present in his article a review of some methods commonly used for this kind of analysis.

" Up to now, most interferograms have had to be evaluated either by reading fringe numbers and their positions manually or by tracing the fringe line by hand with the help of a tracking device such as a graphic tablet. Manual evaluation, however, is a very time consuming and inaccurate procedure."

" Other approches involve the use of image-processing systems to enable a computer-aided evaluation of the real fringe pattern. These systems usually digitize and digitally enhance the fringe pattern. [...] Interactive, computer-based systems allow the implementation of sophisticated algorithms for image enhancement, fringe segmentation, error correction, and fringe numbering, which have to be applied for the evaluation of conventionally observed interferograms."

1.3.2 Modern approaches

When one wants to initiate a scientific project in a given field, the first preoccupation is to look for the things that do already exist in the matter. For this purpose, a close look at specialized books and magazines is recommended, because scientists almost always publish the results of their work.

In the particular field of interferometry, the most interesting literature takes place in magazines such as Applied Optics, Optical Engineering, Laser Focus ... and also in the material designers' pamphlets (like [[zygo80]]). All this can give an up-to-date picture of the state of the art. A careful reading can also provide ideas of fields to explore, parts to ameliorate as well as concepts to keep.

For some of these articles, we are giving hereunder a brief summary of the main ideas and concepts. Their complete reference can be found in the bibliography. It is clear that the list given there and summarized here is not exhaustive and that some articles could have been forgotten. We used all what we could reach.

We are basically interested in the systems using the raw interferogram (the photography of the interferometer's screen) rather than in sophisticated interferometers with built-in analysis devices like those that were presented in [[zygo80]] and [[wyant82]].

In [[haralick83]], the author describes a method for finding ridges and valleys in an image. He also gives a good definition of what is a peak or a valley: they are characterized by the fact that they are a "simply connected sequence of pixels having intensity values which are significantly higher than those neighboring the sequence"

To detect these sequences, he proposes to compute the first order derivative, since at such a high (or low) point, this derivative must have a zero crossing and is then easy to detect. Note that in [[haralick83]], the emphasis is on the peak detection.

In [[yatagay82/1]], the authors propose a semi-automatic system for the analysis of interferograms. Their method is relying on the following steps: the definition the fringe area to be analysed, then the application of three different noise suppressor or fringe enhancement programs, afterwards the extraction of the fringe skeleton and its thinning and finally of an interactive fringe ordering.

Funnell, in his 1981 article ([[funnell81]]), presents fully interactive method

for the analysis of interferometric fringes: first of all a mask has to be defined that will determine the image boundaries, then the user has to define a line cutting every fringe in order to help the program in finding the fringes in the interferogram. When found, the fringes are tracked following predetermined directions. Each time an obstacle is encountered, the program stops and is asking for help.

This small sample of method does in fact represent more or less what can be done in the field of "automatic" interferogram analysis. [[becker85]] also present a review of some methods commonly used for this kind of analysis. He is concluding his article by the mentionning of the main steps of any computerized analysis system:
(1) Digitization of the interferograms and image enhancement, (2) fringe segmentation and fringe coordinate extraction, (3) fringe numbering and correction of fringe disconnection, (4) coordinate transformation, (5) interpolation and extrapolation of fringe number function, and (6) reconstruction of the flow field and conversion of fringe numbers into the interesting flow properties.

This is more or less the scheme that we will follow for our own method. The main difference of our method with respect to the others is consisting in the "intelligency" given to the computer thanks to high level data used to represent the fringes. This will be explained in a formal way in the next chapters.

Chapter 2

OPPORTUNITY ANALYSIS

"A compromise is a solution satisfying nobody."
The political truism

2.1 ESO actual requests in automated interferometry

As described in the introduction and in most of the papers on this subject, interferometry technique is used by the scientists and in industry to test lenses and mirrors. Using the interferograms produced by the tests, one can deduce many conclusions concerning the optical quality of the subject being studied. To interprete the informations contained on these interferograms, one can proceed manually or use a computer-based analysis system, as described by [[becker85]] (see introduction). Anyway, the method to be followed will be something like this:

- photography of the observed fringes
- digitalisation of the photography taken at the previous step (manual operation)
- fringe numbering (manual step) and grabing of the coordinates of the points of the fringes
- analysis of the detected fringes (automated step)

In the future, the opticians would like to dispose over a processing of all these operations, the ideal would be, of course, that the computer self realizes each and every step.

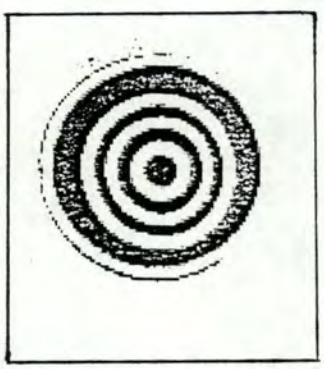
The next chapter will tell us what could be a solution that would lead us to the full realization of such a task.

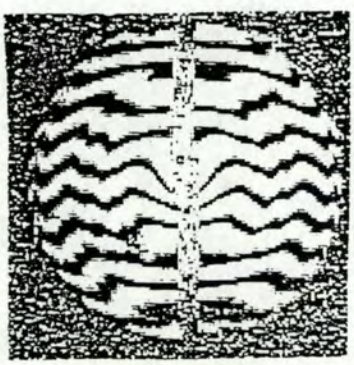
2.2 Design of a global solution for both recording and analysis of interferograms

2.2.1 Introduction : which difficulties will we encounter ?

First of all, we must know that "beautiful" and "ugly" interferograms exist in the nature. The beautiful one look like the one shown in fig. 2.1: regular fringes, good contrast ... while the ugly one is characterized by fringes not straight and/or by holes in the image, hiding pieces of fringes and/or noise limiting the image contrast and/or variations in the thickness of the fringes. An example of an ugly interferogram is shown on fig 2.2.

fig. 2.1 beautiful interferogram
fig. 2.2 ugly interferogram





2.2.2 An idea ...

It is a fact that beautiful interferogram cause less problems[1] for both recording and analysing steps. Therefore, the first idea would be to transform the

[1]For a more complete description of the problems that can occur, see [[dekker85]]. He presents in a little internal note a non-exhaustive list of the problems that a analysis system will have to face.

"ugly" one into "beautiful" one before analysing the image. This way of doing would have as main advantage the modularisation of the processes. Now, let us see how such an "esthetic surgerey operation" applied to interferograms would be possible.

2.2.3 Draw of a solution.

- noise suppress — sometimes, noise appears in the images. It is caused by the devices used to get the interferogram, or in the case of system lenses studies, by the lenses themselves that create sub-interferences that appear in the final image.
- useful part — the useful part of the interferogram is most of the time circular while the photography is rectangular, causing unuseful arrays to appear in the image. The good part must be detected.
- fringe detection — afterwards, the task will be to detect the fringes in the image, that is to extract them from the noisy image environment and to get unuseful signal free fringes. (This means : to have an image where the only visible information is the set of noise free fringes.)
- broken fringes — sometimes, discontinuities may appear in the fringes. to avoid counting the pieces of fringe twice, they must be rejoined. **The aim and main characteristic of this project will be to automatize these edition using high level structures such as tree-structures and and related features**. All this will be explained later on in chapter 4: implementation analysis.
- fringe ordering — using one of the many algorithms that allow us to solve this problem, to follow the fringes and to order them.
- analysis — we may now proceed to the last step of this sequence and analyse our interferogram thanks to the data processed by the previous steps.

2.2.4 Conclusions

Let us call the different steps above : " usefull subsystem" that is, a system which is not complete, but that will be helpfull for the opticians, since it

considers only a subpart of all what could be done for the purpose of analysis of interferograms[2]: fringe recognition and fringe edition.

2.3 One possible useful sub-system

Hereunder, we present four steps that will be part of the system we are planning to build.

⟶ binary	The process consists here into the creation of a new binary image derived from the original one; in other words an image on which only the fringes will appear, thinned and noise free.
⟶ "edition"	rejoining the broken parts of the fringes will be necessary to be abble to order them correctly and to execute the next steps of the processing. This operation will be fully automatic, but in special cases, the operator will have the ability to interactively edit the results.
⟶ ordering	with the binary image obtained and edited at both previous steps, this partly interactive step will create and fill a table with the positions and order number of each fringe detected. The user, in some cases will have to (dis)agree the proposed order and will be given the ability to furnish its own numbering.
⟶ analysis	knowing points coordinates and their fringe order number, we can now execute an analysis step to furnish the values of the optical aberrations or a 3D view of the mirror shape. (See [[zygo80]] and the glossary for the interpretation of the aberrations).

Other methods do really exist, that would certainly provide comparable results. The differences between them often concerns their interactivity degree which has been dramaticaly reduced in our method. The reader will find some of these methods in [[zygo80]], [[funnell81]], [[robinson83]], [[wyant82]], [[yatagay82/2]], [?], [[yatagay82/1]], [[nakadate81]], [[becker82]], [[choudry83]], [[cline82]], [[nakadate83]], [[trolinger85]]. See also chapter 1, section 3.

These articles will be our source of algorithms and methods for our own one. An other kind of approach to this problem is also presented by Becker

[2]see also future developments in chapter 5 to find out the other possibilities or add-ons.

et al. in [[becker82]].

This useful subsystem will consist in:

- the fringe detection (transformation into a binary image)
- the fringe recognition and ordering (rejoining the broken pieces of fringe and give an order number to each fringe)
- the fringe analysis (optical aberrations computations)

The other parts of the whole process are assumed to exist. (Image grabing and fringe analysis).

Chapter 3

FUNCTIONNAL ANALYSIS

"Trouver sans chercher est une chose difficile et rare;
Trouver ce qu'on cherche est commode et facile;
Ignorer et chercher ce qu'on ignore est impossible."
Architas de Tarente.

3.1 Information structuration

This step aims at defining a structure of the informations[1] that are necessary for the realisation of our project.
This organization of the informations will be a static description of each data type's own characteristics *-we will speek of ENTITY-* and will be the expression of the relations *-we will say ASSOCIATION-* that exist between the entities.

When necessary, we will underline some constraints owned by a data type -i.e. : existing constraints and possible values of data[2].

In one word, THE GOAL OF THE INFORMATION STRUCTURATION IS TO HELP PEOPLE IN DEFINING THE SEMANTIC OF THEIR DATA.

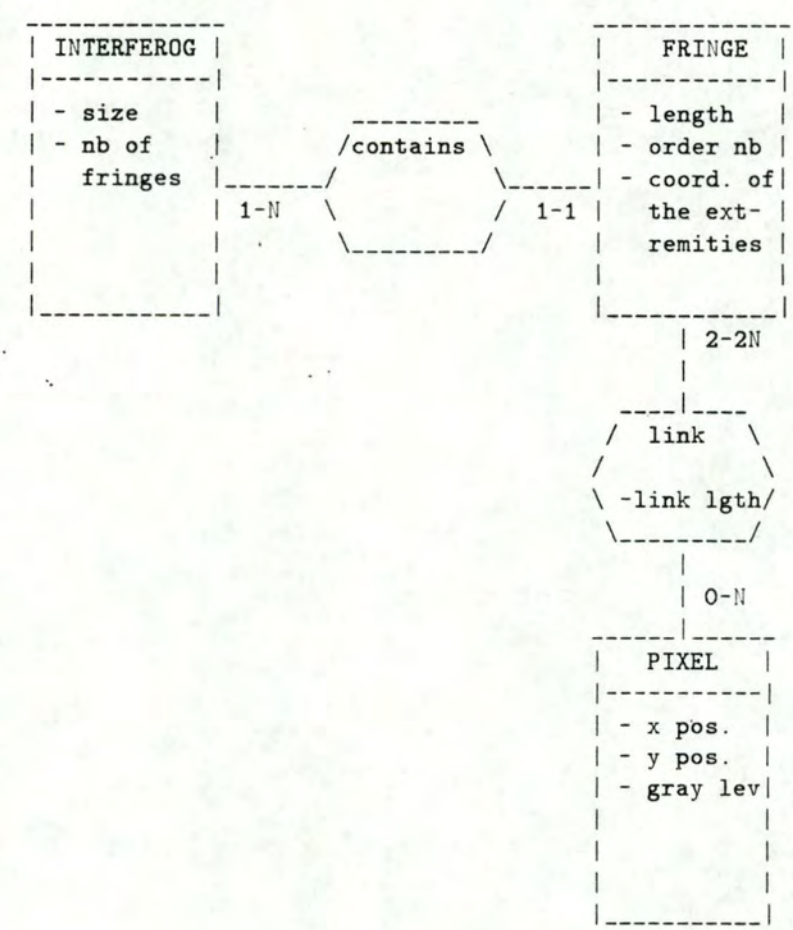
Let us now see which will be the entities and their characteristics on one

[1] "information" means : the potential signification attached to the data.

[2] Example of constraints value: the positions (x, y) must correspond to a pixel somewhere inside the image

side and the associations that link them on the other side.

fig. 3.1 ENTITY -- ASSOCIATION scheme



3.2 Process structuration

This process structuration will give the ability to define a hierarchy, a decomposition of the operations that will have to be executed to realize the project. This structuration will be presented in a static form.

The processes will be structured in 4 different groups that will become as many different levels in the hierarchy :

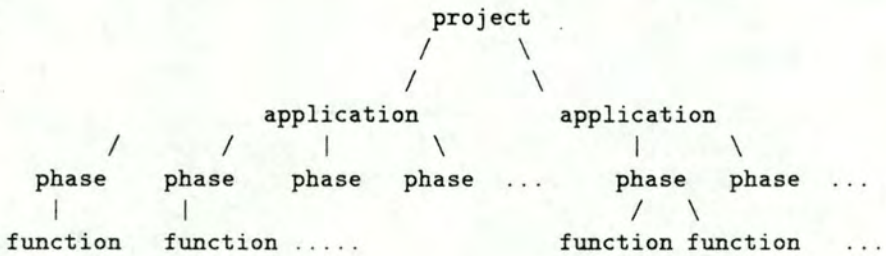


fig 3.2 Process structuration

- The **project** is the object of the analysis. Here, it will be the analysis of interferograms.
- The **application** is a quasi "self-living" process with respect to the other applications of the project. Here, we will say that there is only one application, consisting in the analysis of interferograms. So the project and the application are in this case mingled.
- The **phases** . A phase is a manual or automatic process executed in one location and during an uninterrupted time duration. So, in the system that we have considered in the opportunity analysis, we may consider six main phases.

  1. Photography and disk recording of the interferogram, plus size reduction of the image in order to reduce the execution time.
  2. Filtering and smoothing of the image obtained, plus the definition of its usefull part.
     This part is not mandatory: the operator will have to decide of the opportunity of filtering, smoothing, ... the interferogram, according to his experience.
  3. Transformation of the interferogram obtained at the end of phase 2 in in a binary image. (detection of the fringes)

4. Recognition of the fringes. Automatic step where the disconnected part of fringes are "re-pasted" together, and where they receive an order number.

5. A phase where the user can interactively edit any intermediate result of the process if he judge that it is necessary.

6. The last phase consists in the analysis of the interferogram, whose characteristics are now stored in a table.

Note : the phases that will be considered in this paper will be the phases number 3, 4, 5 and 6 as described in the sub-system of the opportunity analysis, expecting that the others do already exist.

- The **functions**. Each function is associated with an elementary objective.

  The functions of the phase number 3 could be :

  - if we proceed according to Robinson [[robinson83]], the following functions should be defined in our phase 3 :
    1. search for a fringe in the image
    2. search for the orientation of that fringe in the image
    3. search for any local maximum on every line $\perp$ to the line defined by function 2. and set to one the pixel in the binary image corresponding to the local peak found.
  - if we use the Yatagai *et al.* method [[yatagay82/1]], we get something quite different that consists only in two steps :
    1. search for any local maxima in the image and set to "1" the corresponding pixel in the binary image.
    2. thin the so got structures.

    This method will be explained in detail in chapter 4 section 2.1.

  The functions of the phase number 4 could be :

  1. structuration of the points of the detected fringes in order to help in recognizing them.

2. to edit this structure in order to be able to reconstruct the original interferogram (if we are facing corrupted data).
3. to assign a number to each recognized fringe.

The function of the phase number 5 could be :

1. analyse the final data with any method that would use the data processed at the previous step[3].

3.3 Process dynamic

The aim of that part is to show the conditions of the triggering, of the execution and of the halt of the processes in order to indicate the way the system will react and eventually to optimize its run.

Here is the process dynamic diagram :

[3]see also chapter 5, section 2: future developments.

3.4 The processes static

Processes static will give us the ability to better define each process in term of

- the objectives to be realized;
- the quality expected;
- the rules to transform the input into the right output.

This will be specified for each phase and for each function. (Hereunder, we present an example of the static description of our problem at first for the

phases, and afterwards for each function of these phases according to the decomposition decided in the process struturation.)

Phase 3: [—> fringe detection]

- Objective to transform the interferogram given by phase 2 into a new one, similar to the first, but in which the remaining information will be the peaks detected. So it will be a binary image.
- Quality any addition nor rejection of fringes and suppression of the noise.
- Input the interferogram processed by phase two.
- Output a new, binary, interferogram; the previous unchanged.

Phase 4: [—> fringe recognition]

- Objective to edit the binary image given by phase 3 in such a way that the broken fringes be reconstructed and to assing an order number to each.
- Quality automatic, efficient, not too much time consuming.
- Input the binary image.
- Output a table with the characterisitics of the fringes (position of their points and order numbers.

Phase 5: [—> fringe analysis]

- Objective to analyse the data given by phase 4 and to issue a list of all the aberrations of the optic that produced the interferogram. to ensure the availability of data for future application programs.
- Quality the results ought to be proper and accurate.
- Input the table with the fringes points and order numbers.
- Output a list containing the computed optical aberrations.

Now let us consider all the functions defined for each phase:

With the Robinson method [[robinson83]], *we can define :*

Phase 3, Function a: [—> fringe search]

- Objective on the rough image, set a system axes with (0,0) coordinates at random on a fringe of the interferogram.
- Quality easy, fast and efficient search.
- Method search for the maximum on each axis, the maximum must be on a fringe. that maximum becomes the new point (0,0).
- Input the interferogram processed by the previous phase.
- Output the coordinates of one point of a fringe.

Phase 3, Function b: [—> fringe orientation search]

- Objective on the rough image, determine the fringe orientation of which we know a point.
- Quality the result is a line, tangent to the fringe at the given point.
- Method the Robinson method, as explained in [[robinson83]].
- Input the interferogram processed by the previous function; the coordinates of one point of a fringe.
- Output another point coordinate through which the tangent defined above passes.

Phase 3, Function c: [—> search for the peaks on the interferogram and write the binary image]

- Objective search for any peak; set the corresponding pixel in the binary image to 1.
- Quality as few error as possible in detecting the peaks.
- Method the Robinson method : scanning $\perp$ to the line defined in function b).
- Input the interferogram processed by the previous funtion the coordinates of the line.
- Output the binary image; the rough image left unchanged.

If we use the Yatagai et al. method [[yatagay82/1]], *we can define :*

Phase 3, Function a: [—> search for the peaks on the interferogram; write the binary image]

- Objective detect any pixel that can be considered as a local maximum; set to a "high value" the corresponding pixel in the binary image.
- Quality as few error as possible in detecting the peaks; fast; applicable to any kind of interferogram.
- Method we present in chapter 4, section 2.1 of this analysis a complete description of the method. the interested reader will find in Yatagai *et al.* [[yatagay82/1]] an other approach of the description of this method.
- Input the interferogram processed by the previous phase and the parameters.
- Output the binary image and the rough image left unchanged.

Phase 3, Function b: [—> thinning of the fringes]

- Objective to thin each fringe of the binary interferogram.
- Method this function will be realized with one of the Pavlidis method, found in chapter 9 of [[pavlidis82]].
- Input the binary interferogram (the result of the filtering of the original image).
- Output a new binary image.

Phase 4, Function a: [—> positions]

- Objective to get the position of each high-valued pixel in the binary image.
- Quality efficiency.
- Method scanning of the binary image.
- Input the binary image.
- Output the table with the positions of the high-valued pixels.

Phase 4, Function b: [positions —> MST][4]

[4]see next chapter 4 and glossary for explanations about trees and MST.

- Objective to create, with the positions of the pixels given by function a, the minimum spanning tree associated with these data. (See next chapters and glossary for explanations about minimal spanning tree).
- Quality efficiency, speed.
- Method the Rohlf method was used (See explanations in [[rohlf78]]).
- Input the table with the positions of the high-valued pixels.
- Output a new table containing the MST.

Phase 4, Function c: [—> Split]

- Objective at this moment we dispose only of one tree. The goal, now, is to have one tree per fringe.
- Quality efficiency; that is : cut only the wrong links.
- Method do the split whenever the split conditions are encountered [5].
- Input the MST table.
- Output the tables resMST (=MST splitted) and metaMST (=one entry per fringe) (see glossary of terms).

Phase 4, Function d: [—> Merge]

- Objective in order to rejoin the broken parts of a fringe, such a merge operation is necessary.
- Quality efficiency.
- Method create a list of all the possible links; choose in this list the best candidates for merging first.
- Input the tables meta-MSTree and resMST.
- Output the tables meta-MST and resMST edited.

Phase 4, Function e: [—> sorting and numbering of the fringes]

- Objective sort these crossing points, give a number to each of them and give the same number to all the points belonging to the same fringe.
- Quality fast and correct.
- Method sort the points crossing the fringes according to their distance to the first point.
- Input the coordinates of the crossing points.
- Output a table containing x-positions, y-positions and order number of each point of the fringes.

3.5 Conclusions

Now the problem is well defined, we may attach ourselves to the development of all the parts of the problem defined here. The global structure in phases and functions enables us to create our programs in a very modular fashion with all the advantages off such a way of doing : easy to understand, easy to modify or to update, easy to introduce in any kind of software system.

Chapter 4

IMPLEMENTATION ANALYSIS

"Cent fois sur le métier
remettez votre ouvrage."
Proverbe français.

4.1 introduction

This third part of the analysis aims to give a clear and complete description of each phase of the sub-system defined in chapter 2 and detailed in chapter 3. For each function, we will describe as clearly as possible :

- a short introduction;
- the method being used;
- the data structures involved;
- a shematic description of the algorithm.
- a critical presentation of its results : advantages, disadvantages, corrections to apply to the method.

Emphasis will be on phases 3 and 4 : they are those for which we had to create new procedures, to invent method or to experiment new possibilities.

4.2 Full description of the first step : Phase 3

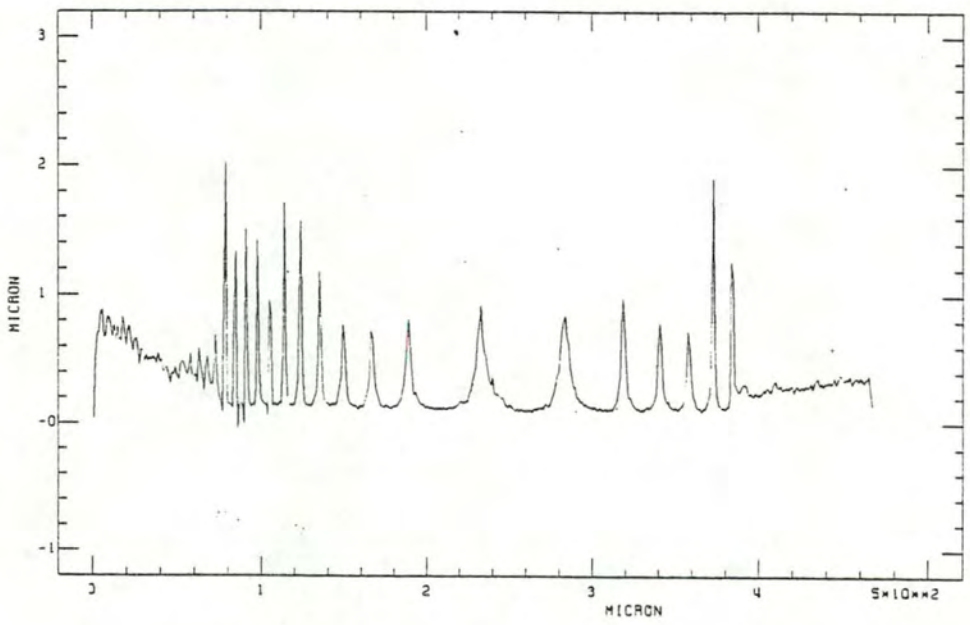
4.2.1 Function a: [—>'search for the peaks in the interferogram]

The method being used.

Because it seems to be the most efficient method for this step, we will retain the Yatagai *et al.* method described in [[yatagay82/1]]. We present here the most interesting part for the realization of this step.

The goal of this phase is to detect the peaks that form the fringes in the interferogram. The problem encountered here is that those fringes are not always well defined and well contrasted : they may become fuzzy along the border of the image; they are sometimes burried in noise or they may appear very broad or very thin, and their thickness is not always constant. So, we have to cope with a huge amount of different cases : good cases, bad cases, ... or worse.

The question is now : in a typical image, what is a peak ? Which are its characteristics ? Let us assume that we have a 1-dimension image (an image that contains only one line). Its general shape ressemble this:



There, peaks can easily be detected since what makes a pixel be a peak, is that it is higher than its neighbors[1] !

If we compare the one-line plot here above with a sine function, for example, we can also remark both peaks and valleys :

[1] In [[haralick83]], the author gives a definition of what is a peak and a valley in an image: a "simply connected sequence of pixels having intensity values which are significantly higher than those neighboring the sequence"

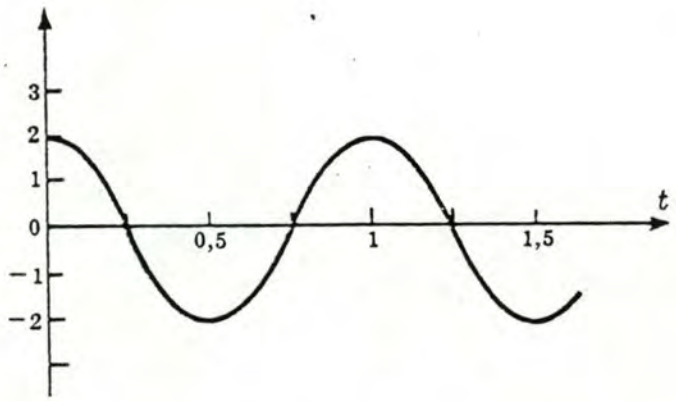


fig. 4.2 a sine function also shows peaks and valleys

The best way to detect these peaks is to calculate the first order derivative and to look for the points in which our derivative becomes zero. This is the basic idea that will lead us to the solution of our problem. But we musn't forget that we are facing two dimensionnal functions, so that we will have to apply the 2 dimensionnal first order derivative (the gradient) on our interferogram.

The Yatagai method proposes a way of computing such a gradient. It uses a window, scanning the hole image pixel by pixel. This window can be either a 3x3, a 5x5 or a larger one. The window size has to be adapted to the fringe thickness of the interferogram. For thick fringes, one has to apply a large window, while thin fringes just need a small one. The center of this window always represent the pixel being tested, in other words, the point in the image for which we try to find out if it is a maximum or not.
The figure on the next page represents such an image scanned by a 5x5 window.

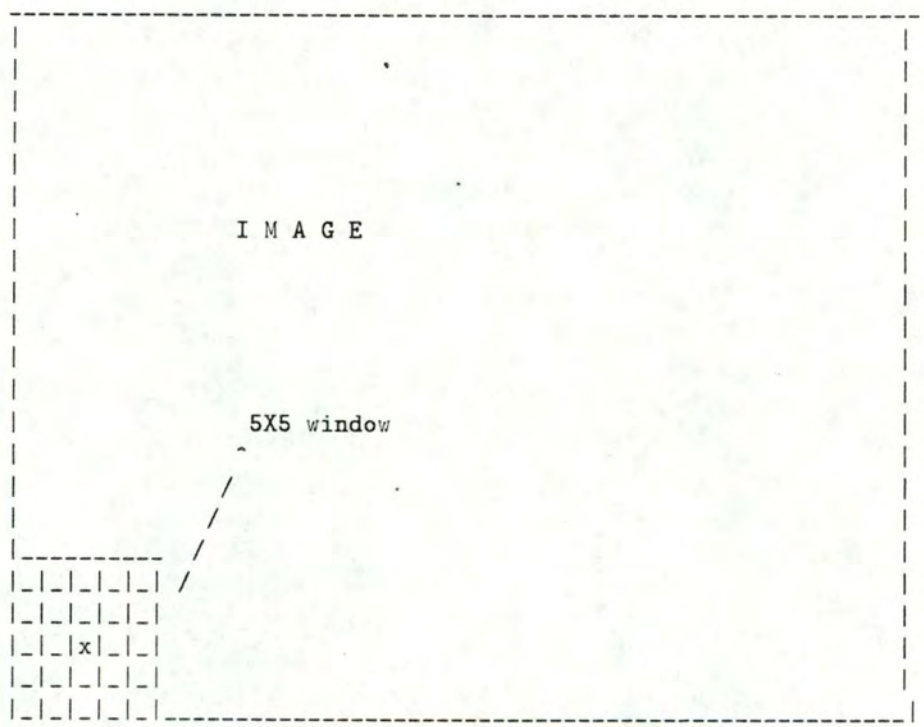


fig 4.3a Representation of an image scanned by a 5X5 window.

Hereunder, we present a view of this window. The XX represent the point being tested.

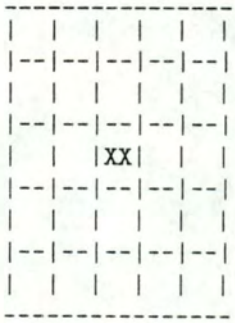


fig4.3b View of a window

The formulæused to compute the maxima were took from [[yatagay82/1]] and have been slightly updated by us (introduction of a threshold to get rid of noise). They represent four different tests to execute. Each of these tests will be a criteria to satisfy. At least one of the four must be satisfied.
We will use the number of criteria to apply : $n = 1, 2, 3$ or 4 of the following tests, according to the level of severity that we want to apply.

1.

$$\sum_{i=-1}^{1} P_{(0,i)} > T \sum_{i=-1}^{1} P_{(-2,i)}$$

and

$$\sum_{i=-1}^{1} P_{(0,i)} > T \sum_{i=-1}^{1} P_{(2,i)}$$

2.

$$\sum_{i=-1}^{1} P_{(i,0)} > T \sum_{i=-1}^{1} P_{(i,-2)}$$

and

$$\sum_{i=-1}^{1} P_{(i,0)} > T \sum_{i=-1}^{1} P_{(2,i)}$$

3.

$$\sum_{i=-1}^{1} P_{(i,i)} > T \left(P_{(-2,1)} + P_{(-2,2)} + P_{(-1,2)}\right)$$

and

$$\sum_{i=-1}^{1} P_{(i,i)} > \left(P_{(2,-1)} + P_{(2,-2)} + P_{(1,-2)}\right) T$$

4.

$$\sum_{i=-1}^{1} P_{(i,-i)} > (P_{(2,1)} + P_{(2,2)} + P_{(1,2)})\, T$$

and

$$\sum_{i=-1}^{1} P_{(i,-i)} > (P_{(-2,-1)} + P_{(-2,-2)} + P_{(-1,-2)})\, T$$

$P_{i,j}$ represent the pixel in the window, characterized by its coordinates. T is here a threshold level expressed in % . It is used to get rid of the noise[2]. This threshold will be efficient, because in most of the cases, the noise is random and almost "flat" while the fringes almost show a gaussian shape.

Below the little pictures explain what is actually computed by the tests we do.

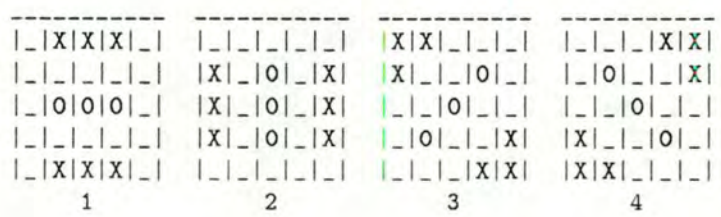


fig 4.4 The four tests presented at previous page and what they do actually represent in the window.

The parameters n and T can determine the accuracy of the peak detection : if n equals 1 for example, too many pixels should be considered as peaks in some cases; if n is too high (3 or 4) the fringes risk to appear as interrupted lines in many locations in the binary image. T can be set to any value (≥ 0), in order to get a good accuracy in the peak detection, and to get rid of the noise.

We musn't forget that the size of the window is a parameter too : it has to

[2]Noise is in this case : local maxima detected between the fringes

be adapted to the fringe thickness.
Another interesting point to underline is that we don't need to define the useful part of the interferogram at this step since the method will consider all what is not fringe as noise and will delete it in the result.

The data structures involved.

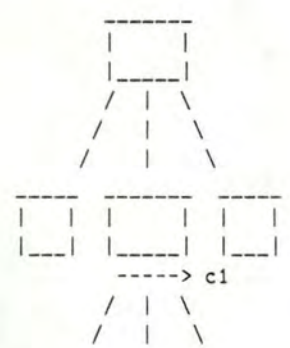
1. INPUT :
   - the rough interferogram [image structure in rows and colums] : (see appendix B for description of the data structures)
   - parameters
     - n number of criteria
     - T threshold value
     - W window size
2. OUTPUT :
   - the binary interferogram [image structure in rows and columns] (see appendix B for description of the data structures)

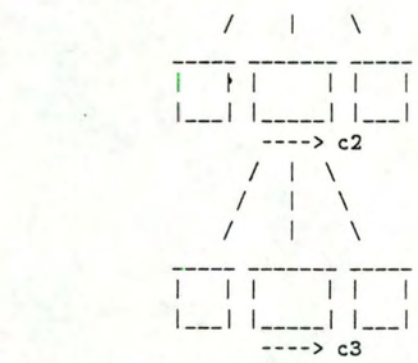
Program design.

1. the calling sequence, with the parameters defined previously should be:

   `FILTER/MAXIMA` *inframe out-bin-frame window-size criteria threshold*

2. program structure






```
c1 : while not eolines
c2 : while not eopixel on one line
c3 : while true_test < n and ntest < 4-n+true_test
```

3. Variables used in this program.

A	1-dimension array that will contain the border of our window
I	the image, considered as a 2-dimensionnal array
BI	the binary image, considered as a 2-dimensionnal array of the same size as I
T	threshold (given parameter) $[T \geq 0]$
W	window size (given parameter) $[W = 3,5,7,9,...]$
n	number of criteria. in fact, the number of tests necessary to declare the pixel in $I(i,k)$ to be a peak. (n is also a parameter)
i,k	indices indicating the position of the pixel being tested in the array I
bl, el	indices of the first and last line of the image to be analysed
bp, ep	indices of the first and last column of the image to be analysed
$true-test$	indicates the number of tests that where successfull for a given pixel. If true-test is greater than n, the pixel is considered as a peak.
$n-test$	contains the order number of the current test (1st, 2nd, 3rd or 4th)
b	contains the maximum displacement in the window according to the window size. i.e. : if $W = 5 \Rightarrow b = 1$; if $W = 7 \Rightarrow b = 2$; ...
sp	variable containing the sum of the "middle of the window" pixels
$s1, s2$	variable containing the sum of the "border of the window" pixels

4. Pseudo-code.

```
Program maxima (inframe,outframe,W,n,T,)

  open inframe
  verify if subframe is part of inframe
  verify n {between 1 and 4}
  verify W {integer,>0,odd}
```

```
    verify T {>0}

    create out_bin_frame ='0
    open   out_bin_frame

    call tests(inframe,outframe,W,n,T)

    close inframe
    close out_bin_frame
    display out_bin_frame
end program.


procedure tests

    do while not end of image
       do while not end of line in the image
          compute the sum of the center of the window
          compute the sum of the edges
          do while nb_test to do < true_test found  and  not maximum
             do the test
          end do
          next pixel in the line
       end do
       next line in the image
    end do

end procedure
```

5. Program code
   See appendix F

Critical presentation of the results obtained.

1. Let us now calculate the time consumation of the algorithm, just to give an example of the performance of this algorithm.
   As a matter of fact, because it processes a matrix (appendix B shows that an image can be represented by a two dimensionnal matrix), we can expect that the response time be in the order of $O(n^2)$ where n is the size of the matrix. In fact, we have the following terms in our problem :
   - $F(s^2)$: for the initialisation of our image. (s is the image size)

- $F(s^2)$: for the scanning of the original image, pixel by pixel (the window path through the image)
- $F'((w-1)*4)$: the transformation of the window structure in a linear structure (see subroutine interface-border in the program code (appendix F))
- $F''(n)$: function depending on the number of criteria (can be considered as a constant)
- $F'(2(w-2))$: the subroutine sum-of-the-border (see subroutine sum-of-the-border in the program code (appendix F))
- $F'(w-2)$: the subroutine sum-of-the-middle (see subroutine sum-of-the-middle in the program code (appendix F))

So, we can write the whole expression that will give us the function of time consumation of our algorithm.

$$F(s^2) + F(s^2) * (F'(4(w-1)) + F'(2(w-2)) + F'(w-2))$$

giving $O(s^2, w)$. The pace of our algorithm is then dependent of the square of the picture size and of the window size.

2. Now, let us look more precisely to the results and to the effect of the application of different parameter values.

   - T the threshold level. It is very useful to suppress the edges and the little peaks that are sometimes detected between the fringes. See fig. 4.5 and following to realize how the threshold can give appreciable results when applied at different values.

   - n the number of tests to satisfy to declare that a given pixel is a peak. The different tests have shown that the most useful values where 2 and 3. 1 gives too much peaks while 4 is too severe. So, we will commonly use 2 as a medium value and forget about this parameter.

   - W the window size (3,5,7,...).
     Let t be the thikness of our fringes (or of the valleys) and s be the size of our image. If t is large, we must apply a large window

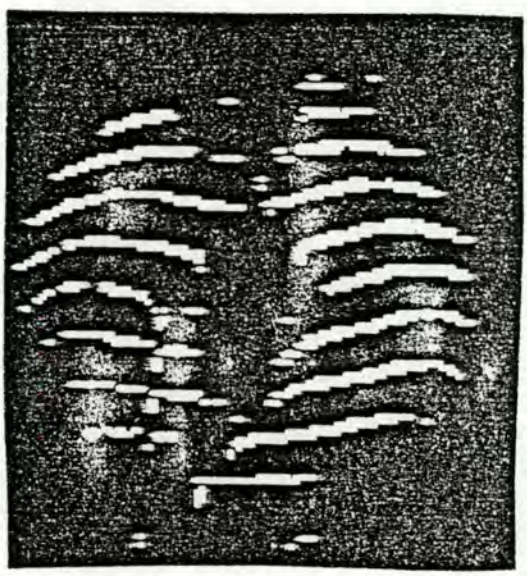
on it. A too tiny window would give us separate points (see figures 4.6a and 4.6b) while a large one provide well defined fringes (figure 4.6c). Suppose now that we enlarge this image, thus we augment s, and t will also augment according to the scale enlargement while W will have to be larger too, in order to keep the same accuracy in the fringe definition. The immediate consequence of this is that the execution time will be highly increased. Suppose now that we reduce this picture size, we can expect the opposite behaviour! With such notes, we may now propose to the user the rebinning of the image before its processing with the filter. The rebin factor (the size reduction factor) will have to be determined according to the number of fringes that one can see in the image, this, in order to minimize the size, while keeping a minimum free space between the fringes (since if there is any free space between them, they are merged together !)

3. The main default of this algorithm is that the window size, parameter defined according to the fringe thickness, ought to vary because of the variation of the fringe thickness among the interferogram ! That's why, for this step, the problem will remain open and new improvements, or a brand new method could be developped in order to cope with this fringe characteristic.

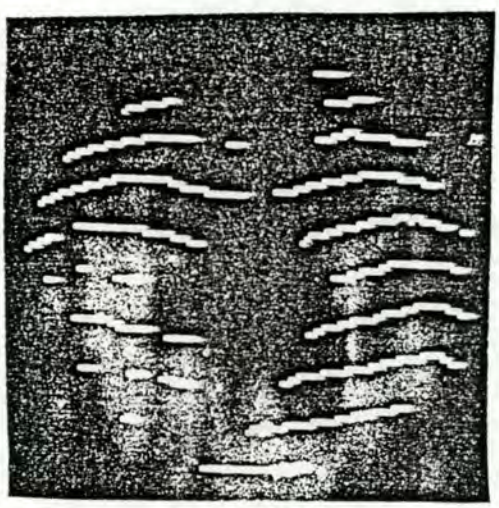
   Another way of doing to keep the present method working in each case would be to filter the same original image with different window sizes and to add the so got binary images. The result of the application of the method with a small window would make appear the thinnest parts of the fringes while the same method applied with a higher sized window would bring us the broadest parts. But this way of doing is quite delicate, and ... longer.

Fig. 4.5 a) original image,
Fig. 4.5 b) image processed with a small threshold level
Fig. 4.5 c) image processed with a high threshold level

a)

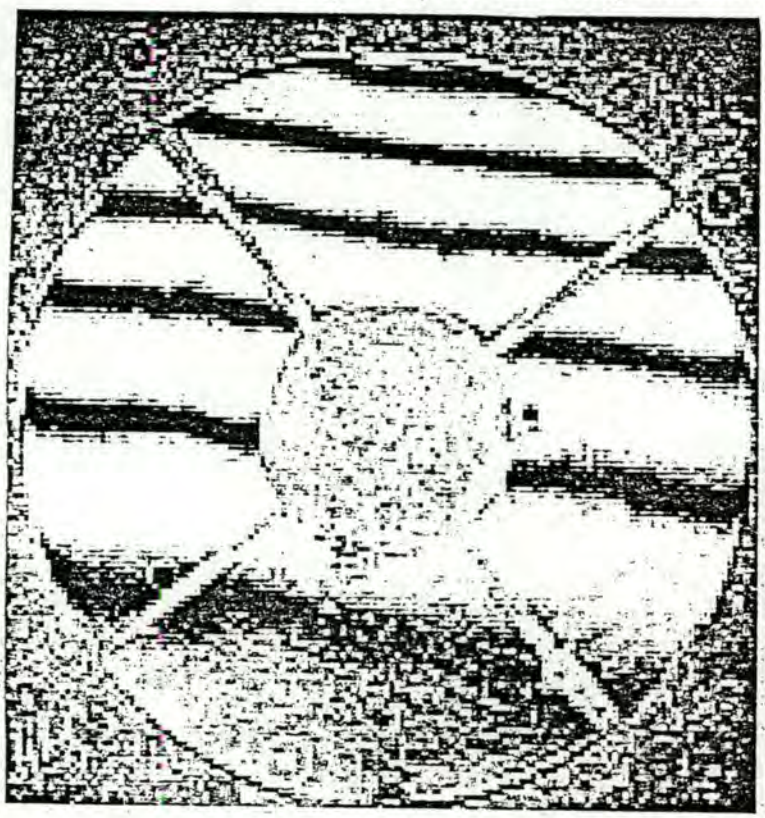


b)



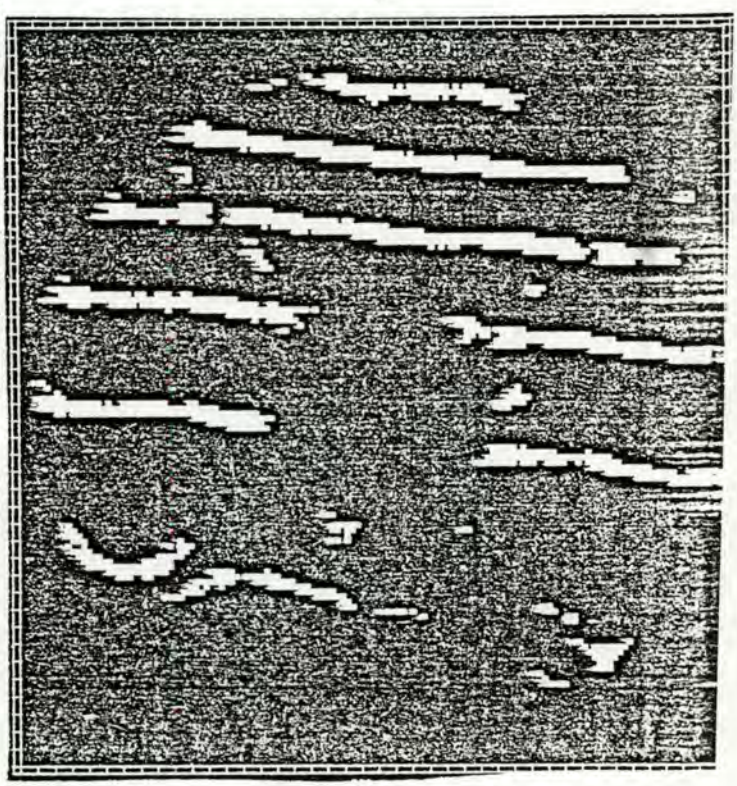
c)

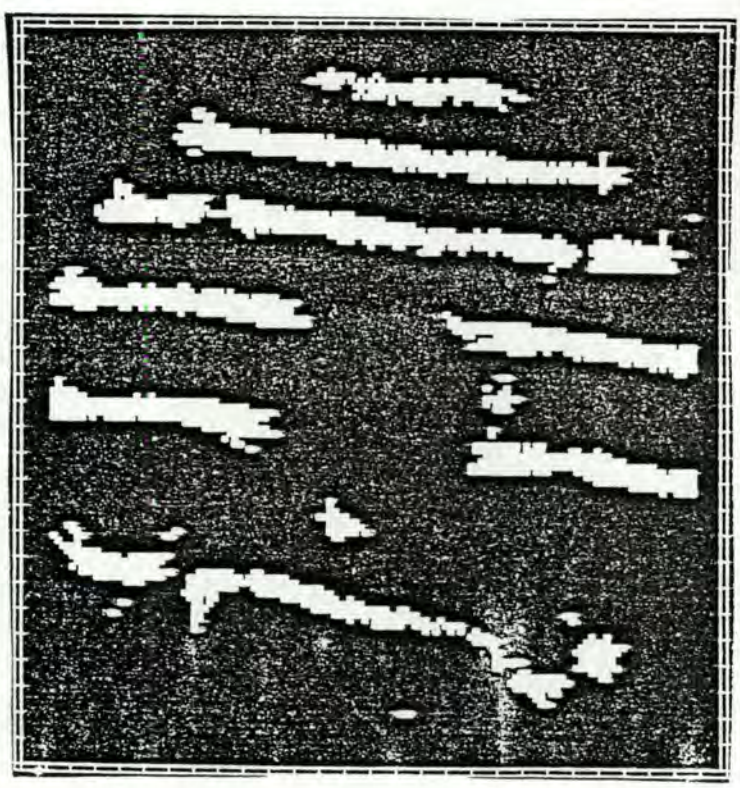
Fig. 4.6 a) original image,
Fig. 4.6 b) the image on which we applied a too small window
Fig. 4.6 c) the same image treated with a large window.



a)

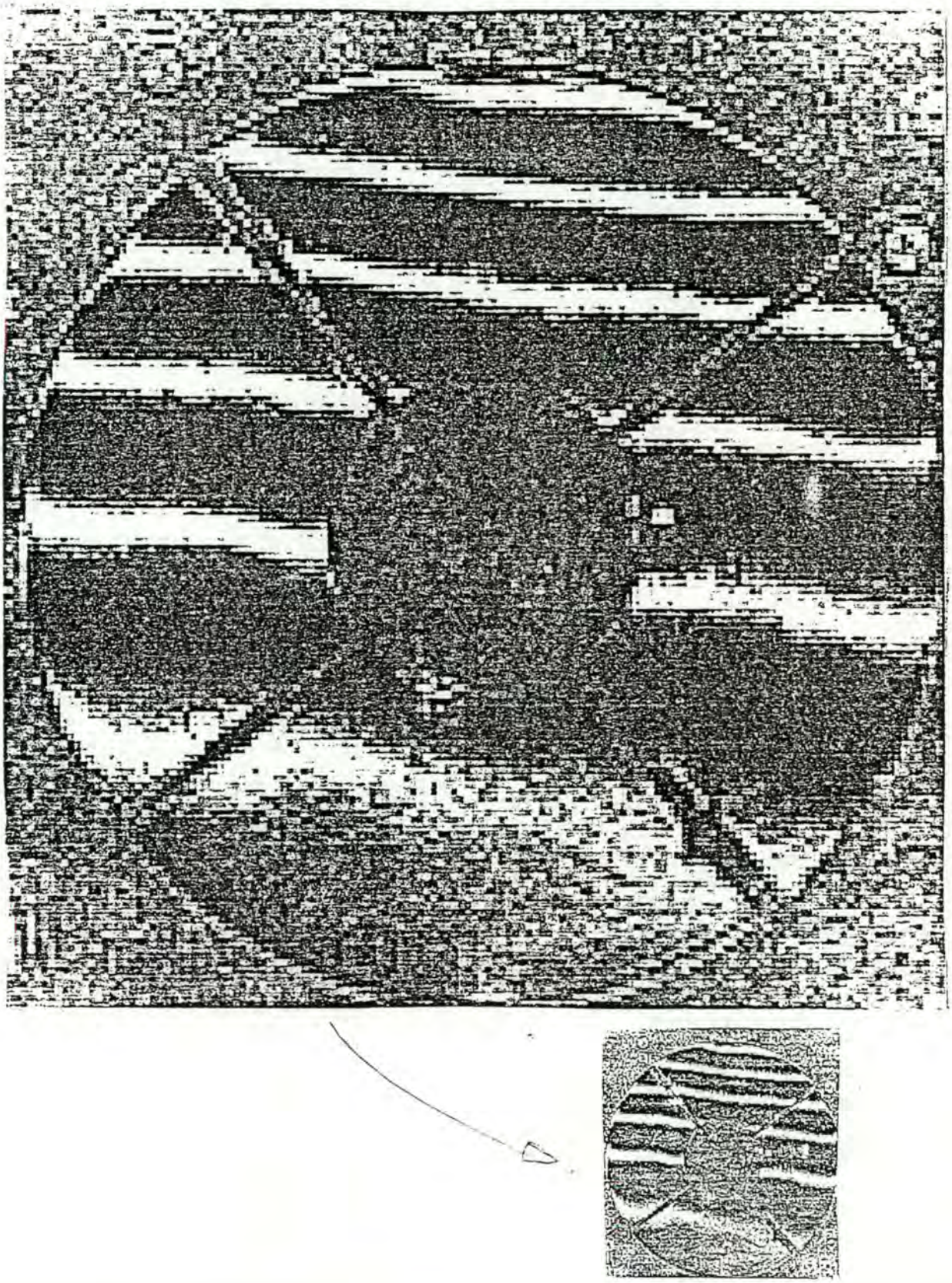
b)





c)

Figure 2.7 in order to speed up the filtering, the rebinning of the interferogram can be applied, keeping in mind that it involves a lost of accuracy and that the fringes must be separated by a least one pixel.



4.2.2 Function b: [—> thinning of the fringe structure]

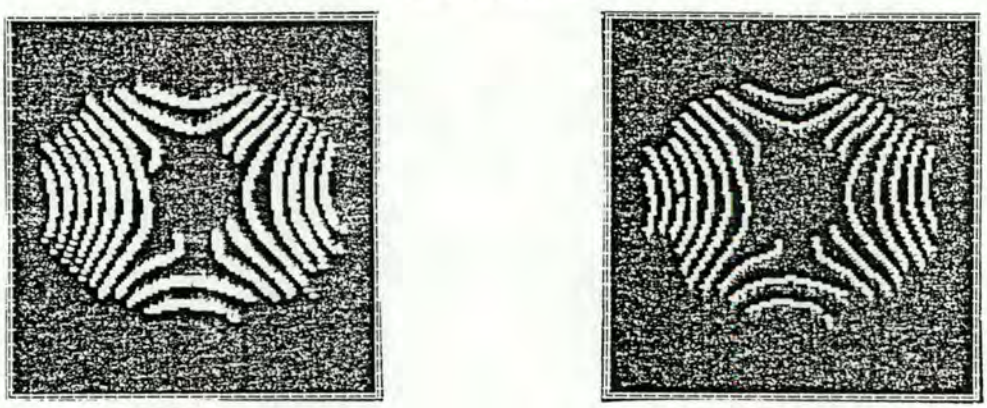
introduction

Why this new step ?
If you remember the figures of the previous pages, presenting the results of the filtering, you will agree with us that the binary fringes so obtained are rather thick ! One consequence of that fact is that we then have many points not always useful, since we are only interested in the center of the fringes and not by what could be called "noise" of the fringes. An other advantage of a so called thinning of the fringes, is that we would get less points while keeping the fringes shape.

description of the method

To thin linear or quasi-linear structures, Pavlidis proposes in [[pavlidis82]] many interesting ways of doing. We have retained only one, which is applicable to our problem and gives proper results. It is called "classical thinning algorithm" and works a way similar to the filtering process described before. As a matter of fact it uses a kind of window to scan the image and flag the points belonging (or not belonging) to the skeleton of the linear structure.

fig. 4.8 here you can see how the thinning works.



The data structures involved

1. INPUT :
   the binary image. (See appendix B for the description of a binary image structure)

2. OUTPUT :
   the thinned image, which is a new binary image but with less points.

3. program design :

   (a) calling sequence :
   `THIN/IMAGE` *bin-image thinned-image*

   (b) the program structure, its variables and its pseudo-code won't be described here since the method and program already exist in the image processing package MIDAS used for this purpose(see appendix A for a complete description of the MIDAS image data analysis system).

   (c) results :
   here above, we present the result of the thinning of the binary image. One can note that the shape of each fringe is kept and that we dispose over the skeleton of the fringes. One advantage of this method is that it requires any parameter.

4.2.3 General conclusions about this phase.

In the case of not-too-bad interferograms (we mean interferograms where the the fringe thickness doesn't vary much), the final result provided by this step is what we actually wanted : to extract the fringes from the original interferogram and to get rid of the noise while keeping the whole information: that is fringe shape and fringe positions.

In some particular cases, if, for instance the threshold level was not set properly, some noise may remain, or too much points might have disappeared. In such a situation the user has the ability to edit the fringes by adding or deleting points using a MIDAS (see appendix A) command and the cursor facility of the video monitor.

4.3 Full description of the second step Phase 4

4.3.1 introduction

In order to avoid the use of too many human interventions in the process of editing the fringes, we have thought of a new representation for these structures that would facilitate their handling and updating.

This new representation is described hereunder. We will also develop more accurately the different functions of this phase because each of these represent different algorithms (easy as well as complicated), which sometimes gains to be carefully explained.

General description of the method.

Remind now that we have got a binary image, processed by the first step of this project. With this binary image, we can write a table in which we represent the positions (x and y) of the high-valued pixels :

fig 4.9 : location of the pixels in an image

In this example, the location of the pixels (marked X in the example above) can be expressed by: $((2,1),(3,2),(3,3),(3,5),(4,4),(4,5),(4,6))$. These coordinates can now be used to create the adjacency graph of these pixels. Moreover, with their positions, the distances between them can be computed.

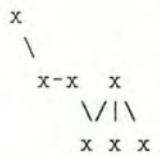


Fig. 4.10 Example of a graph generated with the coordinates of the points of the previous figure.

This is not the graph that we will use. What we will compute, in fact, is the minimal spanning tree (MST) (See glossary of terms for all what con- cerns trees and MST's; see also [[zahn71]] for a complete description of MST's). The MST of our example will be :

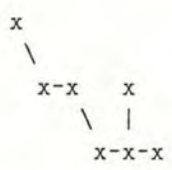


Fig.4.11 The minimal spanning tree graph generated with the points of fig 4.9

because of the shorter distances involved. (The sum of the distances involved in fig 4.10 is higher than the sum of the distances in fig. 4.11.)

The usefulness of such a representation, is that we might edit the tree : with cutting the leaves and little branches, splitting into subtrees, merging subtrees with others and so on. The examples shown at the end of the description of each function will present with more details what are actually leaves, branches, subtrees,...

4.3.2 Phase 4, Function a: [—> table with the positions]

This function aims to get all the high-valued pixels of the thinned binary image to fill in a table with their positions.

Method to get this table filled.

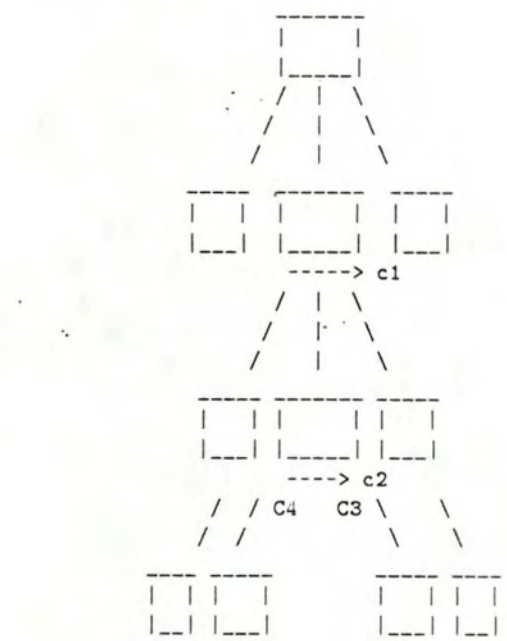
- `scan the binary image;`
- `each time a high-valued pixel is encountered, the current position of the scanner is stored in the table.`

The data structures involved.

1. INPUT
   - the thinned binary image (the skeleton). (see appendix B for a description of the data structures).
2. OUPUT
   - the table with the coordinates (see appendix B for a description of the data structures).

Program design.

1. calling sequence :
   `POSITIONS/TAB` *input-bin-image ouput-table*
2. program structure:




```
c1 : from begin_line to end_line
c2 : from begin_pixel to end_pixel
c3 : high_valued pixel
c4 : zero_valued pixel
```

3. the variables used in this program :

i,k	current pixel position
j,l	boundaries of the image
im, tab	input image and output table
cur-pix	indice of the last pixel stored in the table

4. pseudo-code :

```
i=2
cur_pix = 1
do while i<j
   k=2
   do while k<l
      if im(i,k) = high_value
         then tab(cur_pix,x) = i
              tab(cur_pix,y) = j
              cur_pix = cur_pix + 1
      endif
      k = k+ 1
   end do
   i = i + 1
end do
end program
```

Conclusions.

This little program, easy to write causes any major problem.

4.3.3 Phase 4, Function b: [—> MST]

This function will read the table of the positions and furnish another table containing the MST. (Do not forget to have a look at the glossary of terms to find out what's an MST, if you don't remember the first explanations)

The method.

The method used has been proposed by F. James ROHLF in his 1978 paper ([[rohlf78]])[3].
We have used it because it was reputed to be the fastest at this moment, however complicated.

The data structure involved.

1. INPUT

[3]Rohlf also proposed methods for the clustering of points in an 1972 article: [[rohlf72]]. The reader would also find in [[tanimoto77]] a review of what can be done with graphs and points

- the table with the positions : (see appendix B for a description of the data structures).
- the parameters : a grid size (See method in [[rohlf78]])

2. OUTPUT

- the MST table (see appendix B for a description of the data structures).

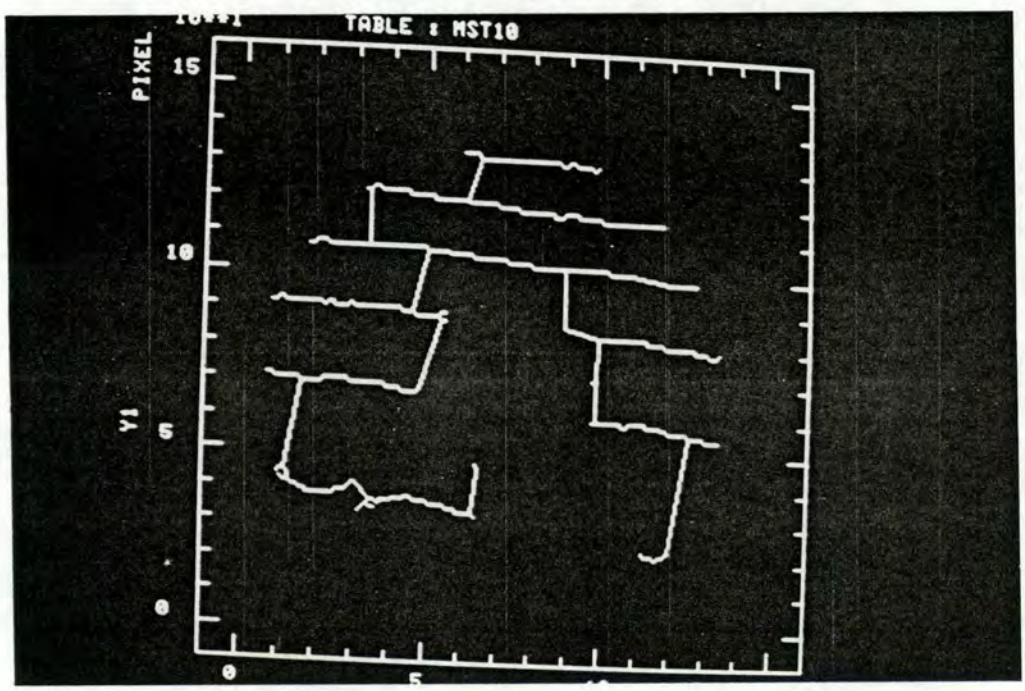
Program design.

1. calling sequence :
   MST/TAB *positions-table MSTree-table grid-size*
2. for the other parts of program design see listing and [[rohlf78]]

Conclusions.

The histogram of the distribution of the distances of the links (distances between the points) gives evidence that points are always linked with their nearest neighbors. This way of linking ensures us that we will keep our fringe structure. The longest links are then obligatory inter-fringe links.

Fig. 4.12a : The MST-table plot showing the links between the different pixels. The shortest links have been created between the closest points, while the longest straight lines represent the links generated to connect the fringes together.



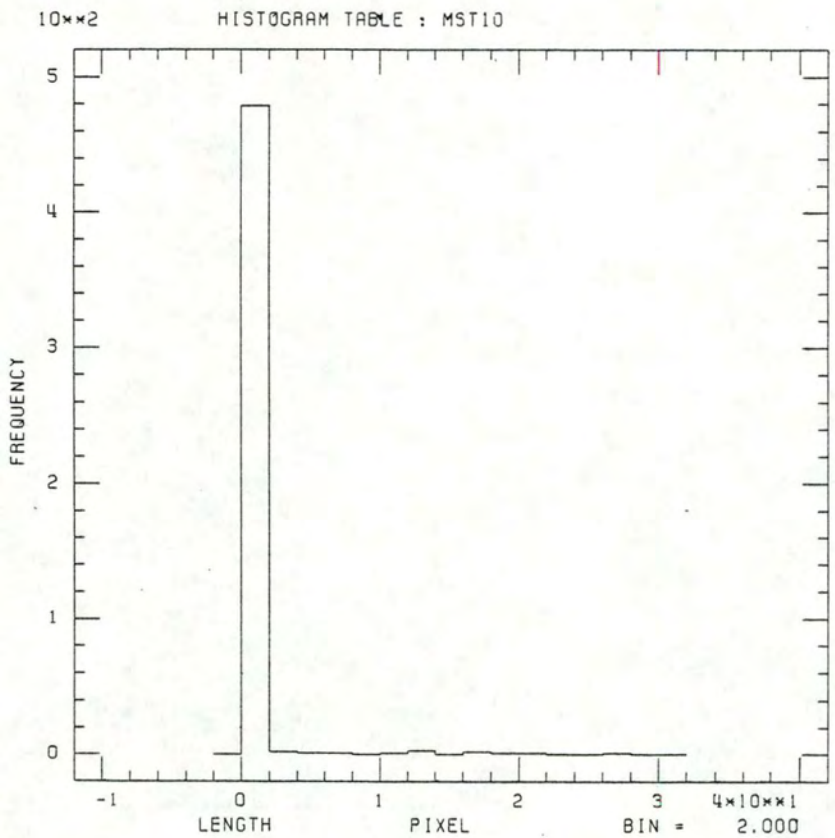


Fig. 4.12b : Histogram presenting the length of links distribution generated in the MST of an interferogram. Note that the most frequent links are short links.

4.3.4 Phase 4, Function c: [—> Split]

Now using the Minimal Spanning Tree that we have created thanks to the positions, we may continue the process and try to split the fringes, try to separate them into single subtrees. Each subtree will be a fringe or a piece of fringe.

Method.

In a first attempt to reach this goal, we will **cut the links** between points that are far enough from each other ($dist > \sqrt{2}$) and, especially, the links that have a too large angle difference with the one formed by the neighboring points.

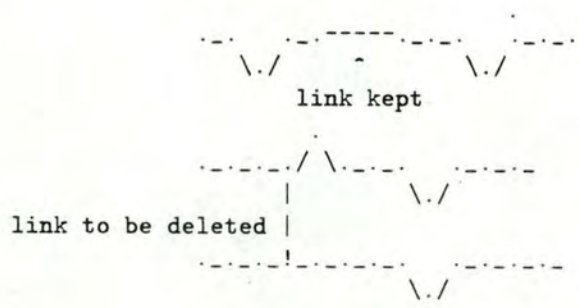


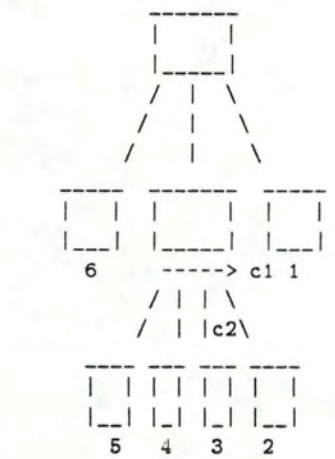
Fig. 4.13 Links to keep or to suppress.

the data structures involved.

1. INPUT :
   - the MST table (see appendix B for a description of the data structures).
2. OUTPUT :
   - the edited MST table witch will contain less entries since the tree has been now splitted. (It means that links have disappeared)
   - the metaMST table witch contains one entry per detected fringe. (see appendix B for a description of the data structures).

program design.

1. program structure :




```
c1 : while not end of tree
c2 : node has more than one son

1 : transform MST into a more conventionnal tree form (see
    variable used in the program for the structure)

3 : compute the tree tilts
    look for the smallest angle difference
    split if angle difference and distance satisfy the conditions

6 : update metaMST giving the fringe length.
```

2. the variable used in this program :

tree	array containing the MSTree transformed in a more usefull way : for each entry, you have (son1-pointer,son2-pointer,son3-pointer,father-pointer, dist-to-father,x-coord,y-coord)
pt	pointer to the current node in tree
tilt1	angle between one point of the link and a point of its neighborhood at a depth it depth.
tilt2	angle between one point of the link and a point of its neighborhood at a depth *depth*.
tilt3	angle between the two points of the link
indi	current indice of MST
indo	last entry of resMST
indm	last entry of metaMST
depth	depth of search in the neighborhood of one point. (useful for the computing of the local angle of the fringe)

3. pseudo-code[4] :

```
          include MST->tree transform
          pt=1
          do while pt not> last elememt of tree
             if element_of_tree(pt) hasn't more than one son
                then do nothing
                else compute the three tilt using depth
                     compute the six angle differences from these
angles
                     look for the smallest ones
                     suppress the two links that have the great-
est angle
                                difference in resMST
                     add an entry in metaMST
                     endif
             pt = pt + 1
          enddo
          edit metaMST giving the length of fringes
       end split
```

[4]For this function, [[aho74]] and [[knuth73]] have been helpful in providing some algorithms for tree scanning

conclusions.

This method has been successful in all the cases tested as you can see in the example shown hereunder. It is also quite fast (only one scan and one rewrite of the MST).

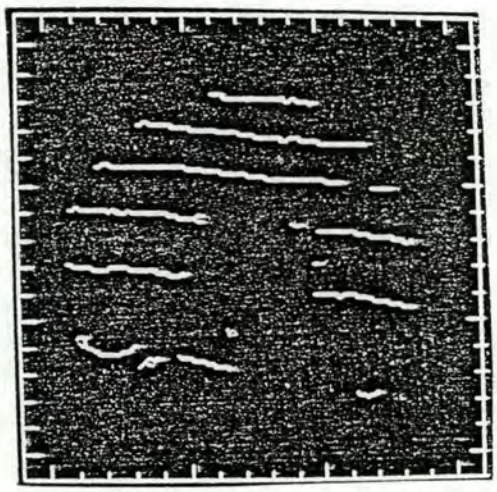


Fig. 4.13b : Example of the application of the split algorithm to the MST: the longest links have been cut.

4.3.5 Phase 4, Function d: [—> Merge]

Now, we dispose over an certain amount of fringes or part of fringes, stored into the MST. The aim of this step will be to merge the part of the fringes not well connected with the best fitting other pieces of fringe or with the border of our interferogram.

Method.

For this purpose, we will build a **list of all the merge possibilities**. That is a list in which each entry will be a reference to an edge point plus a reference to another edge point of another fringe or to the border.
For example : the list of possible links for these pieces of fringes will be the following set of pairs of points :

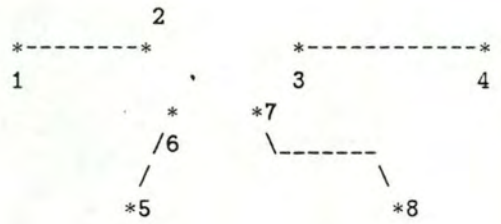


Fig. 4.14 Example of 4 pieces of fringes to link together.

(1,border); (2,3); (2,border); (2,6); (2,7); (3,border); (3,2); (3,6); ...
Note that pairs like (2,5) or (8,4) or (2,4) are not proposed : in fact these points are not in scope of each other.
Each entry of this list will also contain other informations such as the distance between the two points proposed for the linking and a value of the angle difference of the proto-link and the neighborhood of both points on the fringes :

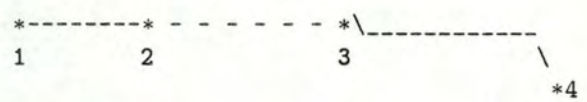


Fig 4.15 How the angles are computed.

proto-link : (link (2,3)) 0^o
neighborhood of 3 : (link (3,4)) -30^o
neighborhood of 2 : (link (1,2)) 180^o

In a second step, our merge process will **search** in the list of propositions **for the best one**, that is :

- the proto-link with the shortest length and
- the proto-link with the smallest angle difference, as explained above.

For example, we will avoid to merge like this :

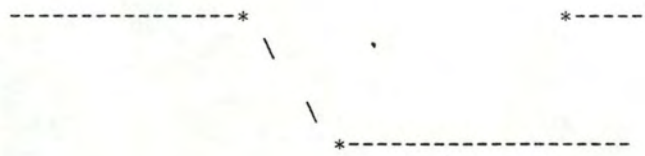


Fig. 4.16a Merge not good : the angles of the proto-link is no the best one.

... and will prefer :

Fig. 4.16b Merge is correct : the angle is better than in fig 4.16a, although the distance between the two points linked is higher.

We will also avoid to merge in this case :

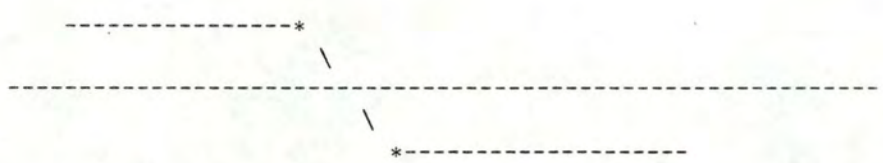


Fig. 4.17 Avoid to create a link that would cross an existing one.

That is, we will have to check if our proposition is not crossing an existing fringe.

The data structures involved.

1. INPUT :
   - the edited Minimal Spanning Tree

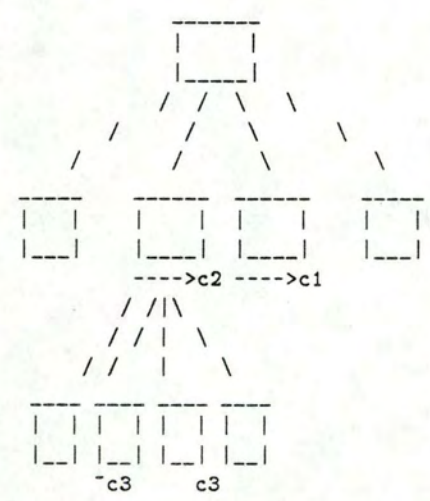
- the meta-Minimal Spanning Tree (see appendix B for a description of the data structures).

2. OUTPUT :

- the MST updated (with eventually more entries sinces new links might have been added)
- the meta-MST updated

Program design

1. calling sequence :
   SPITMERGE/TREE *MST-in MST-updated meta-MST depth*
2. program structure :




```
c1 : while not end of creation of possible_links
c2 : while not end of use of the possible links
c3 : if merge conditions are satisfied
```

3. variables used in this program :
   - tree — array containing the MSTree transformed in a more useful way : for each entry, you have (son1-pointer, son2-pointer, son3-pointer, father-pointer, dist-to-father, x-coord, y-coord)
   - poss — an array containing all the possible links and their characteristics.
   - tilt1 — angle between one point of the link and a point of its neighborhood at a depth *depth*.
   - tilt2 — angle between one point of the link and a point of its neighborhood at a depth *depth*.
   - tilt3 — angle between the two points of the link
   - indi — current indice of MST
   - indo — last entry of resMST
   - indm — last entry of metaMST
   - depth — depth of search in the neighborhood of one point. (useful for the computing of the local angle of the fringe)

4. pseudo-code :

```
create table of possible links
do while not end of all the possibilities
   search for the best angle difference and best
          distance in all the possibilities
   if not crossing  then add a new link to MST
   flag the entries in poss where the two points
          merged appear
end do
```

5. program code : see appendix F

Conclusions

We present hereunder the results of the merge process applied to our MST splitted. Note that we did not add points but *links*. (The points are represented here by small crosses and the links between them by lines.) Note also that one bad case is remaining : two pieces of fringes ought to be linked together since they obviously do belong to the same. Such a case is not a

fault of our algorithm. (The two edge points are not in scope of each other). To solve this problem, we have developped a special program to interactively edit trees. It will be explained in the next subsection.

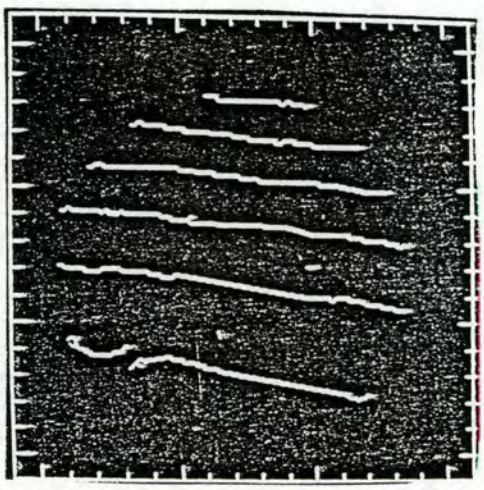


Fig. 4.18 Result after the application of the split and merge process to the MST of an interferogram.

4.3.6 Phase 4, function e: [—> Interactive tree editing facility]

Introduction

In order to correct bad cases of mis-splitting or mis-merging of trees, this editing option has been added. It allows suppression as well as creation of links between points.

Method

The tree is plotted on a graphic display and the cursor is used to indicate which are the points in between which the new link has to be created or the old one suppressed. The program first ask if the user wants to delete, to add links or to quit the program. If the delete link or add link option is selected, the cursor is ready to be set twice near points (for a pair of points). The corresponding link is then deleted (if it was already existing) or created.

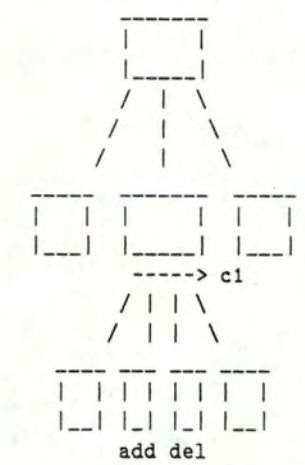
The data structures involved

1. INPUT :
   - the MST
   - the meta-MST (see appendix B for a complete description of the data structures)
2. OUTPUT :
   - the updated MST
   - the updated meta-MST (see appendix B for a complete description of the data structures)

Program design

1. calling sequence :
   MODIFY/TREE *in-MST table out-MST meta-MST*

2. program structure :




```
c1 : while option <> "Exit"
```

3. pseudo-code :

```
ask for Addition, Suppression, or Exit
do while option <> "Exit"
   get cursor position twice
   if option is "Addition"
      then
         add an entry to MST
         if edge points then update meta_MST
      else
         search for a link involving these two points
         if found
            then
               delete this link
               if edge points then update meta_MST
            else
```

```
                display "inexisting link !"
            endif
        endif
        ask for option
    end do
```

Conclusions

Since any method can be reliable up to 100% , it is always necessary to plan specific tool to give the user the ability to react when he realizes that the program doesn't react as one would like it do. That's why this little tool has found a place in the application.

4.3.7 Phase 4, function f: [—> Fringe ordering]

Introduction

At this step, we dispose over well defined and well connected fringes. The aim of this function will be to assign each of them an order number.

Method

The method used is to set two repairs on the representation of the tree on the graphic display and to looked for all the points crossed over by the line defined by the two repairs.

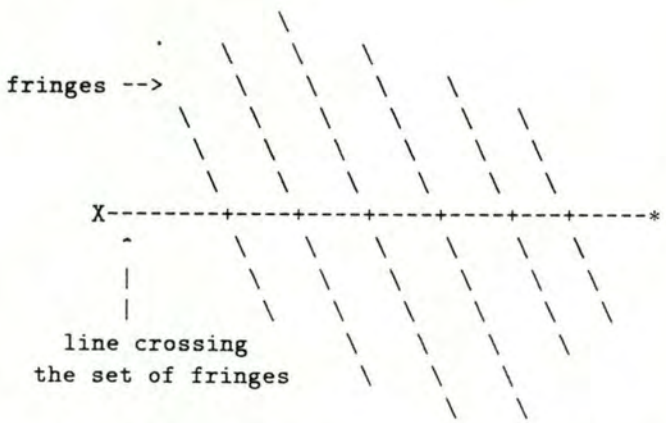


Fig. 4.20 The method used to assign fringe order numbers

These crossing points are then sorted according to their distance to the first repair (for which a first order number was given), and they get ascending numbers. All the points linked to the crossed one get the same order number. That's the way the fringes are numbered.

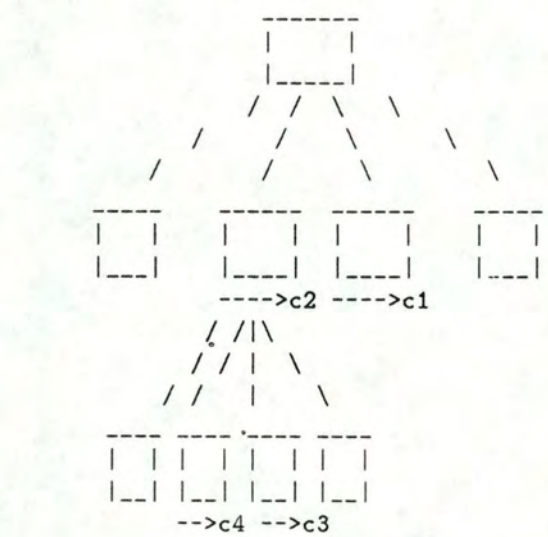
The data structures involved

- INPUT :
  - the MST
- OUTPUT :
  - the table with points coordinates and fringe numbers (see appendix B for a complete description of the data structures used)

Program design.

1. calling sequence :
   NUMBERING/TREE *MST-in tab-in tab-out [fringe-dir first-order-nb]*

2. program structure :




```
c1 : while not end of repairs
c2 : while not end of the repairs
c3 : while not end of links
c4 : while not end of the fringes
```

3. pseudo-code :

```
get repairs
do while not end of repairs
   search for the crossing points
   sort them according to their distance to the
            first cross
   do while not end of the fringes
      do while not end of current fringe
         give the order number to point j of
                  fringe i
```

```
                next point of fringe i
            end do
            next'fringe
        end do
        next pair of repairs
    end do
```

Result.

Hereunder, we present an image, generated from the table created by the numbering function. In this image, one can see each fringe represented in a different color, evidence that each fringe has a different order number.

Fig 4.21 Image constructed from the table with the points coordinates and their fringe order number. Each fringe gets a different color to prove the accuracy of the numbering.

4.3.8 General conclusions about this phase.

This step of the analysis process is what makes our method basically different from all what we could find in the literature.
This way of representing the fringe structure with a high level representation appears to be an original and efficient method, because it allows an easy and comfortable manipulation of these structures (for example in the tree edition).

This type of high level data representation will also be useful in the future if, for example, one wish to build an expert system for the analysis of interferograms : its data could be expressed in term of tree structure[5].

But the fact that it is a brand new way of doing might present some innacurracy when one wants to manipulate them (for example in the split and merge procedure) and this might involve the need for more reliable new versions.

4.4 full description of the third step: Phase 5.

4.4.1 Introduction.

At the beginning of this step, we just dispose over a set of points belonging to the fringes and described by their coordinates and their fringe number. This is enough to apply now procedures for the analysis of the interferogram generated by these fringes.

The only program over which we dispose at this moment is called "WAVE". It is computing the optical aberrations that the fringe shape show. It is clear that other programs exist that would use the same set of data for such an analysis. Up to now, "WAVE" is the only program adapted to our system. In section 2 of chapter 5, other possible data processing method are presented.

4.4.2 Phase 5, function a: [—> Compute aberrations]

Introduction.

Since the development of this program was not requested and was not actually in the scope of our useful sub-system, we have used and adapted the existing program "WAVE" to our system. Its structure is described hereunder.

[5] But does this method not already act as an expert system? See chapter 5 for considerations about this topic.

Method.

- this program will first take a sample over all the points of the table (at about n^2 points if n is the number of fringes).
- then, it will normalise their coordinates : that is, it will include all the points in a circle whose center will be the point (0,0) and its radius will have length 1. (Up to now, the coordinates were expressed in pixel values, the origin being in the image lower left corner).
- the next step is to fit the phase map to a set of orthogonal quasi-Zernike eigenfunctions representing the well known optical aberrations.

Data structures involved.

1. INPUT :
   - the table with points positions and fringe numbers.
2. OUTPUT :
   - a listing containing the numerical results and general informations about the analysis.

Program design.

1. calling sequence :
   ANALYSE/WAVE *table-name sampling-factor, wave-length-path-between-two-fringes, 5th-order-spherical-aber.,mesured-on-the-surface operator-name units*[6]
2. program structure : since this program was almost already existing we re-used it. It is constructed according to the method explained here above.
3. program code : see appendix F.
4. results :
   On the next page we present the table giving all the aberrations. For the interpretation of the results, see the glossary of terms under "aberration".
   (The main aberrations have been underlined here).

[6] All these parameters will be explained in the user's manual in appendix D

```
*****************************************
* W A V E  -  v. 2.0  ESO/MIDAS  09-01-86 *
*****************************************

LABO5 #2            PDS

Analysis does not include 5th order spherical aberration

Date of the analysis run      24-JAN-86
Done by                       BENOIT
Aberration amplitudes are in Nanometer
measured on the surface
Fringe spacing                   316.50

Number of points used               229
```

RMS	79.6	72.6	72.4	14.4	61.1	69.3	14.1	13.6	13.
TILT	948	781	785	902	778	784	900	896	913
TET	-82.6	-86.8	-87.7	-83.0	-87.6	-87.4	-82.5	-82.4	-82.
DEFO	187	119	7	204	87	103	266	277	263
COMA		264	265	53	261	265	53	46	75
TET		-65.9	-61.2	102.3	-62.5	-63.2	115.8	121.8	108.
SA3			94				-52	-65	-41
AST3				191			191	189	196
2TET				-176.3			-176.0	-177.6	-175.
CTRI					96			12	3
3TET					99.9			159.0	-89.
QUAD						60			16
4TET						-61.3			47.

Fig. 4.22 Example of the results produced by the program wave. A list of all the aberrations is presented with, for each of them, a value of the amplitude and of the orientation angle of the defect. The main ones have been underlined there.

Chapter 5

RESULTS and CONCLUSIONS

"La morale de cette histoire ... "

In this chapter, we would like to have a look at three important points. The first one will be to briefly describe the current version of the package and to underline its usefulness and efficiency.
In a second section, we will underline the advantages of the present method over the other ones. The third section will be devoted to the description of some new stuff to add in this toolkit. We will also mention in a fourth section some other fields of application for which the whole (or part of the) package could be used with interesting results. The last section of this conclusion will be devoted to a demonstration of the capabilities of the package: some results will be shown, including the intermediate ones.

5.1 Implementation

All what has been described in the previous chapters was implemented as a set of commands, available in a special context of MIDAS[1]. Each of these commands provides a special function, part of the process. For example, typing

```
FILTER/MAXIMA INTERF BIN 7 2 10
```

[1] see appendix A, description of MIDAS

will extract all the maxima of the image INTERF, an interferogram, and will store the corresponding pixels in an image BIN containing only the values 0 and 1 corresponding to the maxima detected. For this purpose, a 7x7 window has been choosed; two of the criteria (explained in chapter 4 section 2.1) will be used and the threshold level has been set to 10%.
In the same way, the command

```
SPLITMERGE/TREE MST RESMST METAMST 6
```

will reorganize the minimal spanning tree MST to give well separated and/or reconstructed fringes in the structure RESMST. The meta-minimal spanning tree METAMST where each fringe will be characterized by some parameters, necessary to individually identify them and eventually to manipulate them (see chapter 4, section 3.4 and 3.5) has also been built. See next section to discover one possible use of this structure.

Upon this set of so called *level 1* commands, we have built a special *level 2* command to provide a user-friendly interface to the analysis package. This special feature is designed as an interactive procedure that executes in the right sequence all the level 1 commands necessary to realize the complete analysis. With that way of doing, the user is delivered of the typing of an important set of commands and parameters. This program also provides the possibility of backtracking in the procedure and to furnish all the relevant intermediate results, properly displayed. With these results, the user may judge and react immediately.
A full set of tests have been conducted with the users that helped much in designing and tuning this procedure. Now we are facing an effective and efficient tool.

A complete description of all these commands is given in the user's manual (appendix D).

5.2 Advantages among other methods.

With respect to all the known existing methods for the analysis of interferograms, none of them does propose a procedure that allows as fast and accurate analysis of fringe patterns, with low cost investment as the one we present here. As a matter of fact, anybody who wants to have precise and rapid results for his interferogram may come to a workstation with his photography, and run the system. Less than five minutes later, he comes out

with a sheet of paper giving an accurate value of the aberrations presented by its interferogram.
Moreover, the only things he did during these 5 minutes were to set up the camera, to store the image on disk and to run the procedure. Depending on the quality of his image he had nothing or few things to type on his keyboard: if the fringes were not well defined he had only to change some parameters and to rebegin some steps of the process. In the worst cases, he had to trigger the interactive image or tree edition.

To summarize, we may claim that we have created a method for the analysis of interferograms witch seems to be now one of the most efficient ones, tough some improvements are still in developing stage. (See next section)

5.3 Future developments

"A program that is never updated
is a program poeple never use."
The programmer truism.

To transform the current package from an efficient tool into a powerful engine, some add-ons or modifications could be brought.
These improvements could take place in :

- the changing of the fringe detection step of the method in order to cope with fringes whose thickness vary much.
- the split-and-merge process : one could ameliorate the criteria for the linking or for the merging, since this method is a brand new one in the field of fringe recognition.
- the automatic and accurate detection of the boundaries of the interferograms.
- other optical data analysis programs to extract as much information as possible from the fringes and to represent the result of this analysis not only as a list of numerical values representing optical aberrations, but, for example, also as a 3-dimensionnal view of the mirror shape.

As a further enhancement for this method, one could also think of the introduction of artificial intelligence in such a field. As a matter of fact, it is now obvious that more and more fields of applications are open for the development of the so called *expert systems* to help in solving decision or diagnosis problems. The analysis of interferograms includes a step that we have called "fringe recognition" that is, a step where the fringes have to be reconstructed and numbered. Why not giving such a responsability to an expert system that would manage to do this job with accuracy? But is the application that we have built so far from being an expert system? Reading the definition, (in the glossary) it becomes obvious that few modifications would transform these common Fortran programs into an expert system structured package. As a matter of fact, the *knowledge base* can be the set of criteria that we defined in chapter 4 sections 3.4 and 3.5. The *the fact base* is obviously here the points defining the fringes, while the *rule base* describes the way the criteria are applied to select the links to cut or the proto-link to create. The *inference engine* is then the program `SPLITMERGE/TREE`. The tree editing facility can also be considered as the user interface. The *proof supplier* is the only missing module: it would explain the logical path followed to finally choose a given link to be splitted or linked.

5.4 Other fields of application

As explained above, the package is composed of a set of commands, individually useful for any other application. For example, the `FILTER/MAXIMA` can be used on any image presenting one or many bright structures, the `MST/TAB` and `SPLITMERGE/TREE` commands can help in solving problems of the clustering of sets of points and so on... To give actual examples, the "fringe recognition" commands can be used in spectrographic data reduction as explained in [[pirenne85]], while the whole process could be adapted for automated fingerprints identification.

Another field of application, related to astronomical data reduction is currently under study. The aim is to detect, to recognize and to classify galaxies discovered on photographic plates using the method described in this book. The opportunity analysis of this new challenge is described in [[pirenne86]].

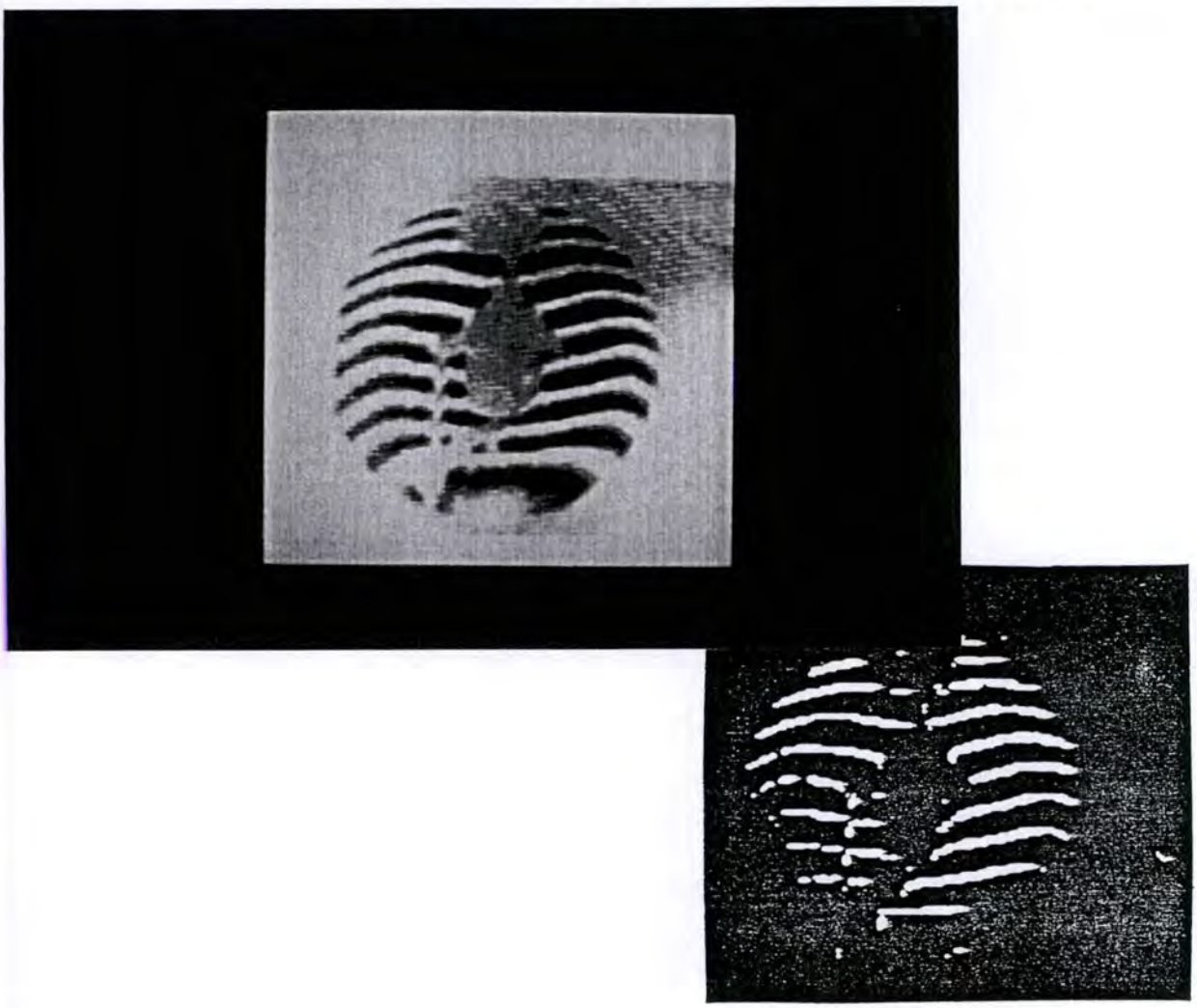
One could also mention that the data structures used to represent parts of an image should find a place to develop and to be used in the electronic movie industry: using such a structure, one could isolate parts of an image

and animate or manipulate them very easily. The same idea could also be used in computer aided education: imagine a student learning human anatomy and pointing parts of the human body on an image display device and moving them to the right place. This is only an example. Only the imagination will limit the field of application of such a data manipulation tool.

The universality of the possible applications of these commands is due to the modular structure of the package, providing re-usable basic tools. This is also obviously the philosophy of MIDAS as you can see in appendix A.

5.5 Results

This section aims at presenting some results obtained with this package and shows the evolution of the process through the intermediate results.



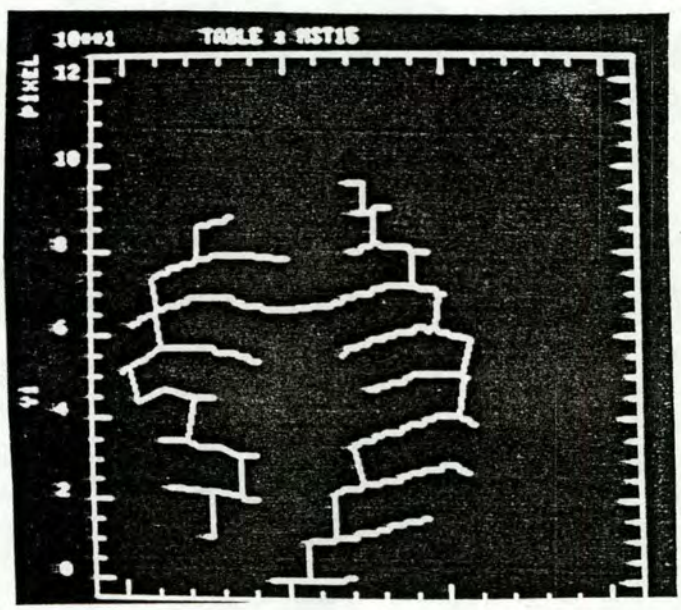


TABLE 2 RST16
PIXEL
Y1

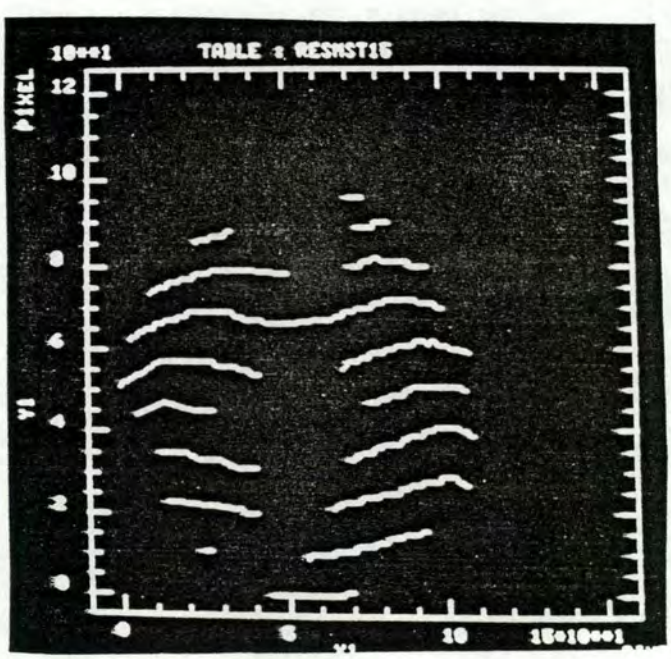


TABLE 2 RESRST16
PIXEL
Y1

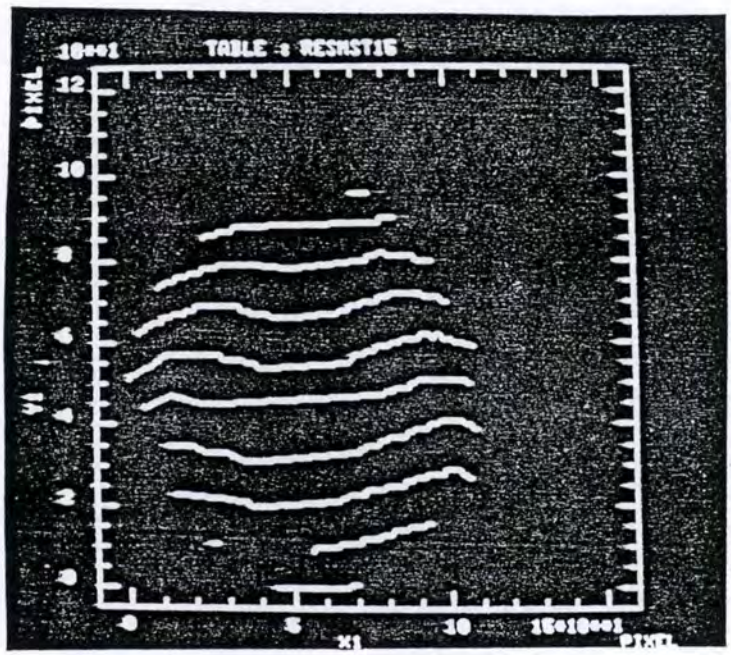


TABLE : RESISTIS
PIXEL
12
10
8
6
4
2
0
5
10
15


D0015
START
0.000
0.000
CUTS

Appendix A

Description of the System Used.

A.1 Hardware.

The hardware configuration currently used at ESO is described on fig. A.1 hereunder. It is basically composed of two clustered VAXes super-mini computers, each supporting a given pool of terminals including image-processing[1] workstations.

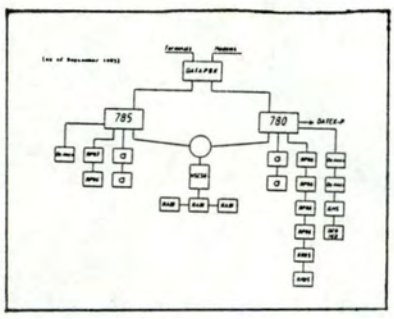


Fig. A.1 The computer configuration currently used at ESO.

[1]The image-processing software used is called MIDAS (see section 2.2)

A MIDAS workstation is typically composed of three different CRTs and of two keyboards. The first keyboard/CRT pair is used to enter a MIDAS session and to provide the computer the commands to execute. This terminal also receive the results of the computations, the on-line helps, the mail messages and the error messages.
The second terminal is a graphic display terminal used to plot the graphics, i.e. curves, histograms and so on.
The third terminal is composed of a high resolution color monitor and of a joystick. The display device is used to show the images while the joystick is helpful to interactively point and define zones to process in the image.
The typical MIDAS workstation can be seen on fig. A.2.

Fig. A.2 A common MIDAS workstation.

The hardware also comprises other typical devices such as disk- and tape drives, many different type of printing devices (line printers, dot-matrix printers, laser printers, graphic printers and so on) and a device especially designed to copy the digital images on photographic film. The system is of course also equipped with modems to allow communication with the widespread world of astronomy.

A.2 Software.

A.2.1 Operating System.

ESO is now using the VMS* operating system on its VAX computers. This is a widespread and quite famous operating system well known by all the Digital Equipment Vax computers users. We won't describe its possibilities and characteristics here since it is not the aim of this work. Interested readers should have a look in [[ipg85]] and in [[vaxprimer]] to get information on this system.

It is now important to note that ESO is currently planning changes into its image-processing package MIDAS in order to adapt this tool on machines running under UNIX*, another operating system primarily developped at the University of Berkeley (California) and now becoming available on more and more mini and micro computers, such an adaptation having as main goal the spreading of MIDAS.

A.2.2 MIDAS – the ESO's Image Data Analysis system.

MIDAS is the acronym for **M**unich **I**mage **D**ata **A**nalysis **S**ystem, because it has been developped at ESO, in Garching near the Bavarian main city Munich since 1979. Let us cite [[adorf85]] to have more precision about this system: "Currently, the MIDAS system itself is VAX/VMS-based, the operating system which is used by about 80% of the European astronomical sites. The MIDAS system itself can run in both *interactive* and *batch* modes. The interactive user can also create batch jobs running in parallel with the interactive work. MIDAS presently contains more than 250 astronomical applications programs. Since 1982 MIDAS has been exported, free of charge, to about 30 other institutes. "

But, what is an image data analysis system?

To give a proper answer to this question, we first must know what is actually an image. Since a "scientifical" definition of this term is given in the glossary, we will just summarize it here.

An image then, is typically something that can be perceived by one of our sense: the view. Thanks to our eyes, sensible to the light, we can **see** things that are illuminated. Many tens of years ago, Daguerre invented the photography to project the reality on paper. Since then, the system has been improved, giving the cinema, the colors, the TV... But these ways of keeping a track of the real world discretize it, not only in time (1 image every 24th of a second for the cinema), but also in space, since a photographic

film is composed of a certain amount of tiny points each characterized by a local intensity and a local color. This implies that the resolution is limited: it means that the image won't be able to be much magnified.
When an image is stored into a computer, the same phenomenon appears: the image is divided in small points called **pixels**. The particularity of the images commonly handled by computer is that they are coded in black and white. To have a color one, you must display three different images, recorded with different filters. Their merging will make appear the colored image.

What is an image data analysis system ?

Let us now turn to the rest of the explanations about data analysis system and answer the question: what kind of analysis can be made on an image? From the previous sentences, we learned that an image is composed of pixels, each pixel representing a given information (position of the point in the image, intensity of light at that point). The whole image (the whole set of pixels) represent another global information. In some cases, this information is not sufficiently clear to be interpreted by the human eye. That's why systems to help in interpreting the contents of images have been built.
These systems at least propose a way of enhancing the contrast by the color coding of the different grey levels of the original black and white image. This is useful since the human eye doesn't easily distinguish between close intensities, while the colors can be well separated. Image processing packages also usually include instructions to manipulate these frames, i.e. zooming effects, extractions of parts of the image, ... and in most of the cases, special features to allow computations on these images i.e. enhancement of the contrast, manipulation of the pixel values, storage in tables and interactions between tables and images. Moreover, they provide high-level command languages to easily perform all these operations.
Let us now turn to MIDAS and discover its structure.

MIDAS' structure.

MIDAS is a *command oriented* software. For example, to display an image on the color monitor you just need to type:

```
LOAD/IMAGE  MYIMAGE
```

where MYIMAGE is an image name. Each command is structured in three parts: the command-name (i.e. LOAD), the qualifier (i.e. IMAGE,or MASK or

CURSOR...) and the parameters (i.e. image-name or table-name). As a matter of fact, one could type:

```
LOAD/LUT  RAINBOW
```

to display the **L**ook-**U**p **T**able called RAINBOW to code the different grey levels of the image currently displayed with the color of the rainbow.
This command language has been built to allow combination of commands (sequentialization, loops,...) in a special procedure program that can be executed either in batch or in interactive mode.

To support all the features mentionned here above, MIDAS has to dispose over many data structures (cited from [[adorf85]]):

- *Images*, which contain a collection of coded pixels;
- *Masks*, containing byte or bit masks which can be used to select an area of interest within an image;
- *Tables*, containing data arranged in rows an columns and not necessarily of the same physical significance. Tables are extremely useful for storing and processing the results of data reduction;
- *Descriptors*, containing the information which is to be physically associated with an image or a table, e.g. its name or the number of pixels on each axis of the image;
- *Keywords*, which are global variables used to provide communication between MIDAS application programs.

Now it is clear that since MIDAS has been designed at ESO, it is a tool especially designed for astronomical use. It comprises several commands for the analysis of star fields, galaxies as well as tools for the analysis of spectrograms. In [[the messenger]], you'll find many comments on these MIDAS applications.

How can one get the images digitized?

The source of informations for an image processing system is of course the images. They can enter the computer if and only if they have the right structure (see appendix B). To get these images structured, many different ways are available. The easiest one is to use a black and white TV camera,

to focus it and to record the image into the computer *after the image has passed through an electronic device called video digitizer.*
Another solution is to use a device called *microdensitometer.* This machine scans very accurately a negative film and store the information on a magnetic tape, computer readable. A third mean is the so called *CCD-camera* now becoming widespread in the public since the introduction on the market of the new video-8 home video system. Such a system directly produces a digital signal from the analog image captured by the camera lens. A last mean that can also be mentionned is the computer itself. As a matter of fact, it can mathematically generate images.

Appendix B

Data Structures Used

To begin this appendix, one could note the following: "every computer program must get structured inputs and outputs, because these programs are not clever enough to automatically extract the relevant data from a given amount of information." For example, if your program is made to compute your budget, it won't provide the expected output if the input data is a Shakespeare's novel ! This is quite obvious since the information structuration is also a guide for the programmer who has to build his program structure according to both input and output data structures.

In our case, 5 main entities are handled by the different programs. Namely: images, position-tables, minimal-spanning trees (MST), meta-MST and trees.

1. Images.
   As described in the glossary of terms, an image can be considered as a set of pixels. These pixels are drawn up in lines and each line usually contains the same number of pixels. The number of lines and number of pixels in the line determine the image size. Such a structure implies that an image will be accessed as a matrix, giving the pixel coordinates of the point to access in the image. For example, one could imagine an image with 4 lines of 5 pixels written like this :

0	2	4	5	4
1	2	5	3	2
3	5	4	2	0
5	3	2	0	0

Such an image would represent a bright line crossing the field from upper right to lower left. This short example will be reused to illustrate the other data structures.
Images are produced in this application by the programs `FILTER/MAXIMA` and `THIN/IMAGE`.

2. The position-table.
In this application, the position-table is used to store all the pixels coordinates recognized as belonging to a fringe. The structure of this table will be:

x, y,fringe-order-number

The fringe order number will be added in the last step to classify the fringes. For example, at the end of the process, the small image here above will produce the folowing table:

x	y	fonb
1	4	1
2	3	1
3	2	1
4	1	1

3. The minimal spanning tree (MST).

This structure is used to store the factual links created between the points belonging to fringes and stored in the position-table. So, for each point of position-table, we store:

- **x1,y1** : the pixel coordinates of the first member of the link;
- **x2,y2** : the pixel coordinates of the second member of the link;
- **dist.** : the distance between the two points;
- **p1** : the number of the link's first member;
- **p2** : the number of the link's second member.

With our example, we can write:

x1	y1	x2	y2	dist	p1	p2
1	4	2	3	1.41	1	2
2	3	3	2	1.41	2	3
3	2	4	1	1.41	3	4

4. The meta-MST structure.
   As previously mentionned in subsection 4.3.4, 4.3.5, 4.3.5 this table does only contain one entry per fringe. For each fringe, we store the following information:

   - **x1,y1** : the pixel coordinates of the first edge of the fringe;
   - **x2,y2** : the pixel coordinates of the second edge of the fringe;
   - **edge1** : pixel number of the first edge;
   - **edge2** : pixel number of the second edge;
   - **length** : length of the fringe (number of pixels);
   - **fringe-number** : order number of the fringe.

   The commands `SPLITMERGE/TREE` and `MODIFY/TREE` produce and update this structure.
   Example :

x1	y1	x2	y2	edge1	edge2	length	fr_nb
1	4	4	1	1	4	4	1

5. Tree structure.
   When all the other structures described above were stored on disk files to be reused by the following steps of the process, the tree structure is an internal use structure. It is build from the MST and is useful for tree manipulation in procedures such as `SPLITMERGE/TREE`.
   A tree is here a table. Each entry is a point that contributes to a link in the MST. For all those points, the following information is stored:

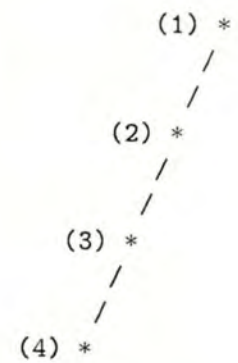
   - **son1,son2,son3** : pointers to the table entry where the first, second and third sons of the current point are stored. Due to the MST structure, each point cannot have more than tree sons.

- **father** : pointer to the table entry where the father of the current point is stored.
- **fat-dist** : distance between the two points that constitute a link.
- **x,y** : pixel coordinates of the current point.

Note : the transformation from MST structure to tree structure is done thanks to a simple interface routine.
In our example : the tree structure of the MST is :

s1	s2	s3	Father	fath-dist	x	y
2	0	0	0	0	1	4
3	0	0	1	1.41	2	3
4	0	0	2	1.41	3	2
0	0	0	3	1.41	4	1

One could represent this tree structure by the following graph :



Summary.

As mentionned above, all these structures are in fact **interfaces** between the different steps of the process. As a matter of fact, only the position-table is actually useful. At the end of execution, one wishes to know the relevant points positions and the fringe they belong to. For this purpose, the original was filtered and the discovered fringes were thinned. The result of this operation was stored in a newly created image. Then, the positions

of these points were stored in the position-table. But to know the actual fringe they belong to, a more "clever" step was created to classify the points, taking into account the possible cases of disrupted fringes. For this purpose, the structures MST, tree and meta-MST were necessary.

Appendix C

Performance Tests (time consummations)

C.1 Why testing the performance ?

They are at least two main answers to this question. The first one concerns the users' requests: it is obvious that the persons who is in charge with the analysis of data wants to dispose over an easy to use, user-friendly and rapid package. Clearly, a computer program whose execution would last longer than a human execution of the same work wouldn't be used for a long time. The second motivation for the performance analysis is that such an operation can help in underlining the most time consumming parts of the process, in such a way that one can eventually cure the problem or at least, propose different remedies. A third advantage of the performance tests is that they can give hints for the extrapolation of the program's behaviour in other conditions: knowing the execution time characteristics, one can easily evaluate the time consummation evolution with other data sets. An example of such a computation is given in subsection *"critical presentation of the results obtained"* of section 4.2.1. There, an evaluation of the limit behaviour of the `FILTER/MAXIMA` program is given, indicating that time consummation is essentially a function of the square size of the image and of the window size applied.

C.2 Some results.

In table D.1 hereunder, we present a list of command CPU/time consummations for 3 images whose different characterisitics are given below.

Oper. \ Image	Image 1	Image 2	Image 3
FILTER/MAXIMA	28"37	25"72	18"56
THIN/IMAGE	2"23	2"96	2"08
POSITIONS/TAB	3"15	3"50	3"03
MST/TAB	11"92	57"18	9"13
SPLITMERGE/TREE	3"90	22"79	2"84
NUMBERING/TREE	1"02	2"83	0"93
+ image size rebinning + 4 images to load on the screen + plot the trees on the graphic display			constant
image size	141 * 143	166*173	128*128
filter size	13	5	7
nb of useful pts	534	2242	458
nb of fringes	6	21	4

C.3 Conclusions.

According to the examples of table D.1, we can note that some programs such as `FILTER/MAXIMA`, `MST/TAB` and in some cases `SPLITMERGE/TREE` are the most time consumming steps. Now, we also know that the higher the number of fringes, the higher the image size, because we must at least keep a two or three pixel distance between two neighboring fringes: this means that if you have few fringes, the image size can be dramatically reduced, reducing then the time consummation of the filter maxima and of all the other steps. So the difficulty is to find the compromise between the maximal image size reduction and the minimal distance between the fringes.

Then, we can conclude by saying that *the* parameter responsible for the global performance of the whole package is the *number of fringes* of the interferogram.

Appendix D

User's manual

"rien ne sert de courrir,
il faut partir a point."
Jean de la Fontaine.

D.1 Introduction

This chapter provides the basic information necessary to understand and to operate the automatic interferogram analysis provided by MIDAS. It includes the description of the whole method as well as its implementation and structure, in such a way that unexperimented users could reduce their data easily.

The package provides a set of different *level 1* commands useful in the case of step-by-step analysis and one *level 2* command especially designed for the one-step analysis.

The method described here is general enough to cover a wide range of typical interferograms. In the present version, the main difficulties will be encountered while trying to analyse an interferogram presenting large thickness variations along their path. Future releases will consider this problem.

The section 2 will briefly describe the algorithms used for the interferogram analysis, i.e. the original image filtering, the fringe thinning algorithm, the fringe recognition algorithms, the editions options ... The description of the operation will take place in section 3, while section 4 will include a brief description of the data structures used.

D.2 Analysis Method

D.2.1 Global structure -' input data - output data

Similarly to the manual process, we have defined the following structure :

- Photography of the image
- definition of fringe center coordinates and their associated fringe number
- analysis of this interferogram.

The *image* of the interferogram will be the input of the process; the *list of the numerical values* of the optical aberrations will be its output.

The package presented here will take in charge the whole process except the photography and the digitization of the image, since some improvements in this step will take place in the near future.[1]

D.2.2 Thin structure - the related algorithms.

The thin structure

The global structure presented here above can be decomposed in the following way :

- definition of the fringe center coordinates and their associated fringe number.
  - fringe detection
  - fringe recognition
  - fringe numbering
- analysis of the interferogram
  - fringe analysis

[1] This is rather a hardware problem.

Fringe detection.

The aim of *fringe detection* is to define the positions of the fringes from the original image. This step is relying on two different subparts[2].

1. The first one, the filtering, consists in the fringe extraction of the image, that is the *maxima of intensity*[3].
2. The second one will be used for the *thinning* of the extracted structure

Filtering.

For the filtering, we will use a window (P) to scan the whole image. For each new encountered pixel in this image, we will have to test if it is a maximum or not. For this purpose, we will use the number of criteria to apply : $n = 1, 2, 3$ or 4 of the following tests, according to the level of severity that we want to apply. In this example, for the simplification, we used a 5X5 window. Let us assume that $P_{(0,0)}$ is the pixel located in the window center and that he is currently being tested.

1.

$$\sum_{i=-1}^{1} P_{(0,i)} > T \sum_{i=-1}^{1} P_{(-2,i)}$$

and

$$\sum_{i=-1}^{1} P_{(0,i)} > T \sum_{i=-1}^{1} P_{(2,i)}$$

2.

$$\sum_{i=-1}^{1} P_{(i,0)} > T \sum_{i=-1}^{1} P_{(i,-2)}$$

[2]In order to speed up the process, an image size reduction pre-processing has been added to the method.

[3]Note that intensity minimum can be detected by inverting the fringes (negative of the interferogram).

and

$$\sum_{i=-1}^{1} P_{(i,0)} > T \sum_{i=-1}^{1} P_{(2,i)}$$

3.

$$\sum_{i=-1}^{1} P_{(i,i)} > (P_{(-2,1)} + P_{(-2,2)} + P_{(-1,2)})\, T$$

and

$$\sum_{i=-1}^{1} P_{(i,i)} > (P_{(2,-1)} + P_{(2,-2)} + P_{(1,-2)})\, T$$

4.

$$\sum_{i=-1}^{1} P_{(i,-i)} > (P_{(2,1)} + P_{(2,2)} + P_{(1,2)})\, T$$

and

$$\sum_{i=-1}^{1} P_{(i,-i)} > (P_{(-2,-1)} + P_{(-2,-2)} + P_{(-1,-2)})\, T$$

Where $P_{i,j}$ is the symbol used to represent the pixel of the window whose coordinates are (i, j). T is here a threshold level expressed in % . It is used to get rid of the noise[4]. Below the little pictures explain what is actually computed by the tests we do.

[4]Noise is in this case : local maxima detected out of the fringes that is, where there's no fringe.

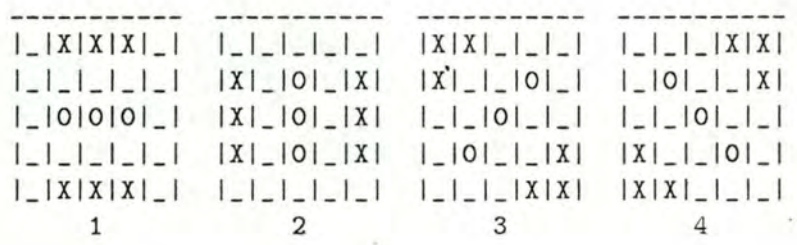


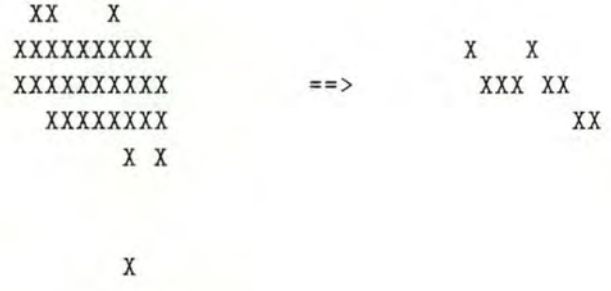
fig. 1

Thinning.

Let us now turn to the second subpart of fringe detection : the thinning of the structure extracted by the maxima filtering. The image presenting the discovered maxima shows "thick" fringes, while what we need is just their *skeleton* that is, the center of the fringe. And this is true, since we are only interested in the fringe shape and not by the width.

To extract the skeleton from a particular structure, the thinning algorithm that we choosed was developped by Theo Pavlidis in his (1982) book "Algorithms for Graphics and Image processing". Interested readers should have a look into it since it is presenting in detail more than one possibility. To describe it briefly, we could say that it also uses a window to scan the image and detect the skeletal pixels by testing the neiborhood of the center of the window.

Hereunder you can see the result of the application of the thinning to two different kind of structures :



```
 XXX
XXXXX        ==>        X
 XXX
  X
```

fig. 2

Note also that the width of our thinned structure will be at the very most one pixel, all along the fringe path.

fringe recognition

The object of *fringe recognition* is to determine to which fringe a given pixel should belong, and so, if such an aim can be reached, to attribute a number to each of the fringes.

For such a purpose, involving computer intelligence, we have defined the following subparts :

- *store the coordinates* of the maximal-and-skeletal pixels into a table.
- use these coordinates to *generate a minimal spanning tree*, in other words, to classify these points into *clusters*.
- *separate* these clusters and *reconnect* those who should be linked together; those who obviously belong to the same set, but are separated on account of a particular and accidental reason.

We will not detail here how the storage of the pixel's coordinates into a table is done, considering that it is really clear; but we will develop more accurately the minimal spanning tree creation, since it is what makes our method different with respect to the other method.

First of all, let us formally define the minimal spanning tree (MST for short).

MST creation.

An MST is a tree. A tree is a graph. A graph is a set of points and a set of ridges, or links between these points. These links can be characterized by a so called *weight*, that can be, for example the expression of the distance between the two points that it relates.

Fig. 3a : a graph with its points and ridges

A tree is a graph with any odd point and no circuits. That is : each point (or node) is connected to the rest of the structure, without introducing loops. One may also say that there always exists a path from any point to any other in the tree. If the ridges are oriented (it means that they can only be followed in one direction), we can also speak of the root of the tree and of its leaves.

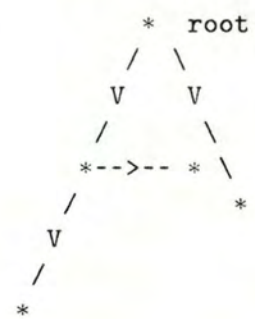


Fig. 3b (left) non orinented links
Fig. 3c (right) oriented links

A minimal spanning tree is a tree in which, the sum of all the links' weights is minimal among all other trees that can be created.[5]

[5]Interested readers are invited to have a look to the nearest math book speaking about graphs or consult M^r Zahn's article about this subject

$$\forall T_i \in \mathcal{T}_P,\ \exists T_j \wr \sum_{k=1}^{n} w_{jk} \leq \sum_{k=1}^{n} w_{ik}$$

where :

- n is the number of links (or the number of points - 1)
- w_{ik} is the weight of link k in possible tree i
- $\mathcal{T}$ is the set of all the trees that can be generated with the set of n points P.

If we can produce such an organization of our points, it is obvious that our fringe structure will be kept, since all the points will be connected (linked) to their nearest neighbors.

Fig. 4.a Points of the fringe
Fig. 4.b The MST created with the fringes of fig 4.b.

Note also that all the clusters (all the fringes in our case) will be linked together, but with a longer link than between the points belonging to the same fringe.

Split and Merge.

Now we have to split the tree in order to separate the fringes from each other and we have also to "re-paste", to merge all the pieces of fringe that obviously seem to belong to the same.
We have called this subpart : the *split and merge* process. The criteria used to split are the following :

- obviously, we will have to cut the longest links, those who connect the fringes together,
- and also the links clearly perpendicular to both fringes that it connects. (See figure 4.b)

The criteria used to merge pieces of fringes are :

- set a new link between pieces of fringes whose extremities are in scope of each other

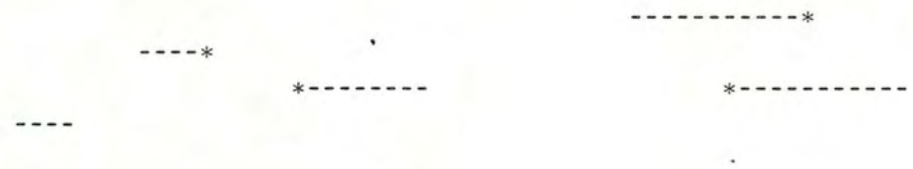


Fig. 5a : Extremities are in scope of each other
Fig. 5b : Extremities are not in scope of each other.

and

- prefer to set a new link between pieces of fringes that do not involve a large angle difference between their respective tilt. In other words, we will prefer the new link of figure 6.b than the new link of figure 6.a.

Fig. 6.a (left) we will try to avoid a link in this case. Fig. 6.b (right) we will try to link in this case.

and

- we will prefer to create a new link between fringe edge points whose distance is minimum.

Fig. 7.a and 7.b : We prefer the new link that has the same angle that both pieces of fringe that it connects.

and

- we will refuse to create a link between two points if that link crosses another one.

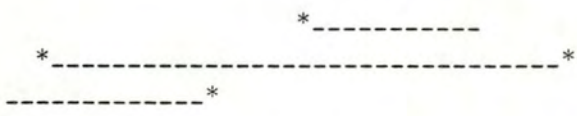


Fig. 8 : avoid to cross an existing link !

Fringe numbering

Now we dispose over a well formed set of separate fringes which must get an order number. For this purpose, the method used will be to set a line over all the fringes in order to cross all of them (fig 9.)

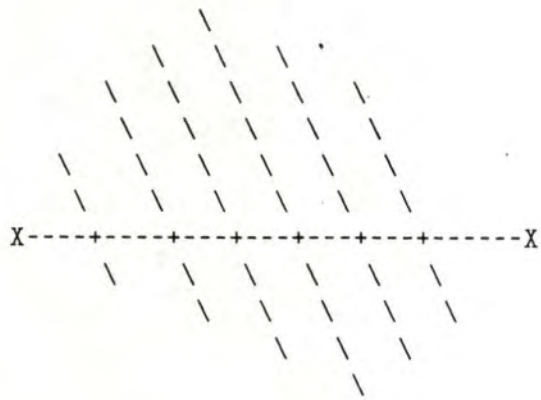


Fig. 9 : the method to assign fringe order numbers.

The order number will be assigned to the fringes increasingly from the beginning of the segment (defined by the user) until its end in such a way that this assertion is verified :

$$\forall f_i, f_k \in F,\ \forall p_{ij} \in f_i \ :\ p_{ij} = n,\ p_{kj} = m\ \ \forall j = 1,...,\mid f_i \mid \quad k = 1,...,\mid f_k \mid$$

$$and\ \ p_{ij} \neq p_{kj}\ \ \forall i, k$$

where

- F is the set of the fringes
- f_i is the fringe i in F
- p_{ij} is the point j in fringe i
- n and m are different fringe order numbers

This means that all the points of one fringe will get the same order number defined for that fringe.

Fringe analysis

This subpart is dedicated to the description of the method used to compute the optical aberrations presented by the fringes we recognized thanks to the previous steps. It involves the description of the program `WAVE`. Since

the author has just adapted the existing feature to run it into the MIDAS environment, this method description subsection will be left blank in this preliminary version. A future release of this document ought to complete this part in explaining how WAVE works.

D.3 Operation

D.3.1 Introduction

In the general introduction to this user's manual, we have seen that two different possibilities are available for the user. The former is easier than the latter, since the former is a program that executes interactively all the operations necessary for the analysis of an interferogram. Little has to be known by the user for its good execution. The latter oblige him to have – at least at the beginning – his user's manual beside him during the process. Moreover, this second method will be longer, because the user will be obliged to manually enter the commands and their related parameters in the right order. But in particular cases, it will be useful for the processing only some of the steps. For example in the case of partial analysis or of application of part of the method to other fields than interferometry[6].

D.3.2 Shematic execution diagram

On the last page, we present a diagram that explain what does actually the first automatic possibility and what should be done if manually proceeded.

[6] see also "other fields of application" (i.e. : PCD frames reduction) in the report.

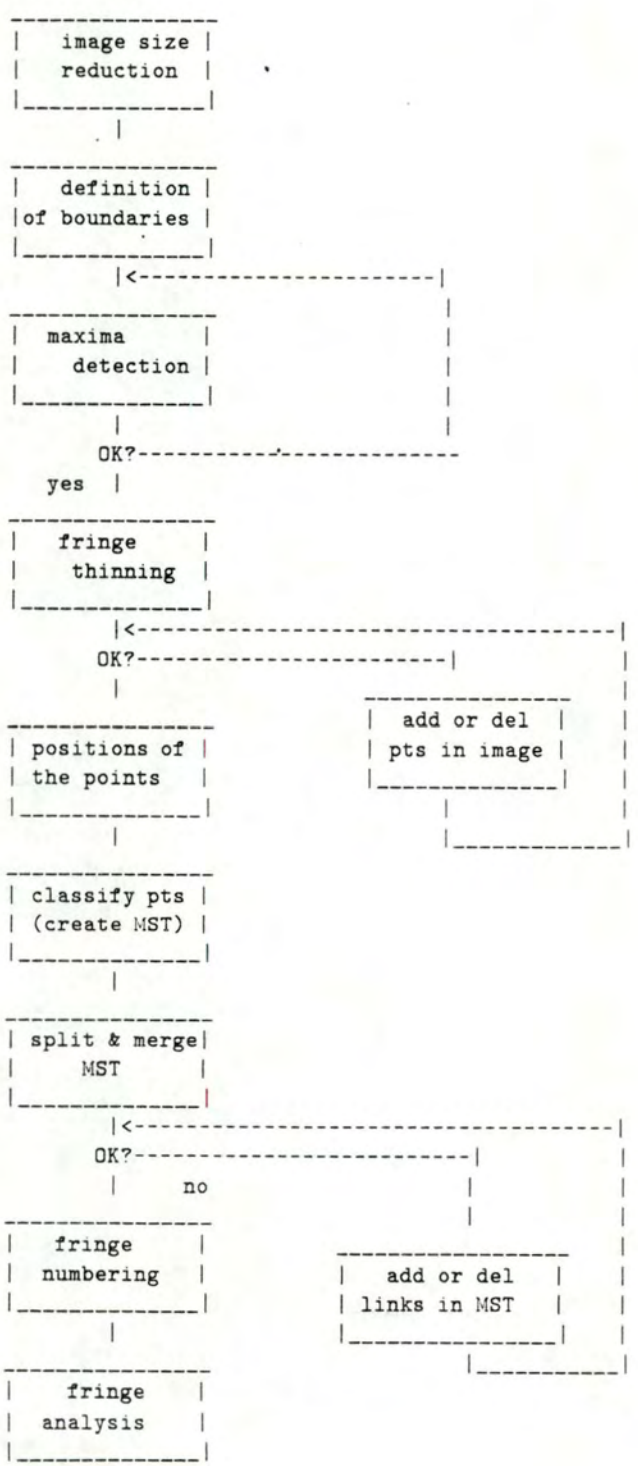


image size reduction
definition of boundaries
maxima detection
OK?
yes
fringe thinning
OK?
positions of the points
add or del pts in image
classify pts (create MST)
split & merge MST
OK?
no
fringe numbering
add or del links in MST
fringe analysis

D.3.3 The level 2 ANALYSE/INTERFEROGRAM command

This one-command analysis has been called "level 2" command because it's managing the sequentialization of the whole set of level one command described in the next subsection.

As you can see on the schematic diagram, this command doesn't include any type of image digitization[7]. This preliminary step will be described in the last subsection of this section. The user that would be in a hurry could have a look there first, and come back here when he is delighted !

At present, we suppose that the interferogram to be analysed is present as an image file in the user's directory and that he has properly initialized his MIDAS session[8]. Typing :

> `ANALYSE/INTERFEROGRAM` image-name table-name nb-of-fringe [fringe-shape] [direct. order-nb] and

his parameters[9] will invoque the main program that will be executed according to the shematic diagram described above.
Let us now describe them :

[7] See reasons in the general introduction.

[8] Be sure you are in the context optics. To make sure type SET/CONTEXT OPTICS.

[9] See also help pages for more particular details.

image-name	the name you gave to the image representing your original interferogram.
table-name	the name of a table that will contain points coordinates and their fringe order number.
nb-of-fringe	the number of fringe contained in the interferogram. This parameter will be used to determine the best-fitting size reduction factor in order to give fringe with always approximatively the same thickness.
fringe-shape	this not mandatory parameter may have three different values to define the general shape of the fringes : • S if they are rather straight lines, • M if they present some slight curvature, • C if they are closed. This parameter will be used only for the computation of the local angle of the fringe near its edge. The default value is M
direction order-nb	if given, this pair of parameters will be used to number fully automatically the fringes. **direction** can hold the values UP, DOWN, LEFT or RIGHT, while **order-nb** will give the number that the UPper, lower, LEFTmost or RIGHTmost fringe will get. If these parameters are omitted, then the cursor facility will be used for the numbering.

This level 2 program will display the results of all the important intermediate steps and will give the ability to backtrack if the user judge that the process has not achieved the expected results.
It has also been designed to facilitate the definition of some parameters (for example in the maxima detection) by computing the best-fitting rebinning factor for each image, providing then a limited choise in the search for the right parameters.
The final results (the numerical values of the aberrations) are issued on a `ASCII` file called `FOR002.DAT` that can be printed on any printing device as well as on the screen. Moreover, the `ANALYSE/INTERFEROGRAM` command

displays the final results on the screen and ask for their printing on the nearest printing device.

D.3.4 Operation with the level 1 command

These commands will have to be executed in the sequence described by the execution diagram here above if one wants to execute them for the analysis of inteferograms. Their parameters and syntax will be explained in the next section as well as in the copy of the "help" pages.

D.3.5 How to store an interferogram image on disk ?

For this purpose, they are two methods currently available at ESO. The former, the fastest one, is to use a photography of the interferogram at to simply put it under the TV camera. The latter uses the negative of the image and the PDS digitizer.

With the TV camera

To operate it, you pwer up the TV camera digitizer and wait for two minutes. In the mean time, just type :

```
> CLEAR/DISPLAY
```

and when the digitizer is ready (making some sharp noise) type :

```
> GRAB/IMAGE
```

then put your image under the TV camera lens, then you need to focus, to set the right contrast ... and so on.
When you think the result is right, just type "ctrl C" and then use the

> GET/IMAGE *image-name*

command to store it in the file named *image-name*. Then, you just need to fill in the descriptor of the image like this :

> WRITE/DESCR *image-name* IDENT "... your comments ..."

The main advantage of this way of doing is that the results are fast. Its disadvantages are that you must carefully tune the digitizer as well as the camera and that the quality of the result is rather poor[10].

[10] non-linearity of the TV camera response, distorsion, overexposure ...

Using the PDS

You just operate the PDS as its manual requires you to do, and when proceeded, load the tape file generated in your directory. The main advantage of this method is that your results have the quality of the original photography! (no more noise or distorsion is added). But its main disadvantage is that it takes much time to be executed. But PDS can run at night...

The best (future) solution ...

... will be to use a CCD camera that will add the qualities of both previous methods, without their disadvantages.

D.4 Implementation

In this section we describe the different data formats as well as their different occurrences related to interferograms analysis, Both images and tabular informations are used to store immediate results of the commands. The section includes also a summary of the analysis commands.

In the command description, we use upper case letters for fixed parts of the commands, while names in lower case are variable parameters. Optional parameters are enclosed in square brackets. Intermediate files are enclosed between quotes.

D.4.1 Images

`INTERF-NAME`	image file, of any size 512X512 if generated by the TV camera. containing the original interferogram. The rebinned image will get the same name !
`'BINARY'`	image file, with the same size as the rebinned original image containing only the maxima detected by the filtering operation.
`'THINNED'`	image, with the same size as both previous one, containing only the skeleton of the structures of image `BINARY`.
`'INTERF'`	generated image, with the same size as the previous images, containing the results of the numbering : that is each fringe with a color corresponding to its order number.

D.4.2 Tables

`TABLE-NAME` name of the table where to store the results of the numbering : that is points coordinates and their corresponding order number.

`'MST'` name of the table containing the results of the classification of the points into a minimal spanning tree structure. That is : the coordinates of the two points being linked, and the distance between them.

`'RESMST'` name of the table used to store the results of the split and merge process applied to the MST. The structure is the same. Only the number of entries might vary.

`'META-MST'` name of the table used to store the results of the split and merge process applied to the MST. The structure will be : one entry per recognized fringe; for each entry: coordinates of both edge points of the fringe plus length and number of the fringe.

D.4.3 Other files

`FOR002.DAT` is an ASCII file used to store the listing of all the numerical values of the aberrations computed for this interferogram. It can be printed as well as typed on the screen.

D.4.4 Commands

The level 1 commands will be :

> FILTER/MAXIMA	interf-name binary window-size nb-criteria threshold
> THIN/IMAGE	binary thinned
> POSITIONS/TABLE	thinned table-name
> BOUNDARY/TABLE	tablename enlargement-factor
> MST/TABLE	table-name mst grid-size
> SPLITMERGE/TREE	mst resmst metamst depth
> EDIT/TREE	resmst table-name resmst-edited
> PLOT/TREE	mst
> NUMBERING/TREE	resmst table-name table-name-nb [direction order-nb]
> ANALYSE/WAVE	table-name sampling-fact,fringe-space,5th-order, user-name units

The level 2 command will be :

> ANALYSE/INTERFEROGRAM	interf-name table-name nb-of-fringe fringe-shape [direction order-number]

All the parameters are explained in the "helps" joined for each command.

D.4.5 keywords

CIRCLE	array of 3 single precision numbers to store x and y coordinates of the circle that contains the fringes as well as its radius. This keyword is filled in by the the a small interactive program using the screen cursor called > BOUNDARY/IMAGE.
POSITIONS	array of up to 80 real elements to store pairs of coordinates of the points used to define the numbering line crossing the fringes. (Up to 20 pairs of crosses might be set).
NUMBERS	array of up to twenty integers containing the fringe order numbers associated to the positions of the crosses stored in the keyword POSITION. (Its first element contains the number of pair of crosses that have been defined).

Appendix E

Glossary of terms

"Il n'y a rien de si utile
que d'envelopper les gens
de paroles inutiles.
J. Prévot.

- **3D-view** A 3D-view is a three-dimensionnal representation of a two dimensionnal image. It means that the image plane is represented in perspective and that the pixel intensities create the third dimension. Such a representation, also called perspective view gives surprising results. See also [[pirenne83]].

- λ Greek letter symbol representing the wavelength of the light used in the interferometer. For example in the Twyman-Green interferometer used at ESO, the laser produces a light beam at $\lambda = 633$ nm, in the red part of the spectrum. (The visible spectrum extends from 780 nm (deep red) to 380 nm (violet)).
  When one speaks of a precision of $\frac{\lambda}{20}$ concerning optics, it means that the most important aberrations for this component have an amplitude lower or equal to $\frac{\lambda}{20}$ of the radiation wavelength used to test the aberrations. Such a value ($\frac{\lambda}{20}$) can be considered as good.

- **Aberration** Aberration means the set of defects presented by optical systems. They involve loss in distinctness in the image produced. Some of them can be cited:

  - *coma* produces extended images (a point gets the shape of a coma)

- *defocus*
- *spherical aberration*
- *astigmatism*
- *triangular aberration*
- *quadratic aberration*

- **Analysis** Analysis is a scientific step that tries to lead to the comprehension of the subject submitted to a test.

- **Association** An association is defined by a correspondance between two or many entities where each assumes a given role. One wants to record information about this correspondance. An association can own attributes. Ex.: the association "own" can link the entities "pupil", "pencil" and "eraser".

- **Batch** Batch processing means, in the computer environment, the ability given by the system to the users to submit jobs without having to wait for their completion and without having to interact with them.

- **Binary image** In our case, a binary image will be composed of pixels presenting only two different states: black or white. These images are useful only to show parts of the image that have been declared as to be relevant.

- **CCD-camera** A CCD-camera is an electronic device used to directly transform the light flux comming on its cells into a numerical value, in such a way that one can directly get computer-ready information. This information is immediately useful and "pure" (it musn't pass anymore through electronic noise-generating components).

- **Cluster** A cluster is a "geographical" gathering of points: that is, a region of space where the density of points is higher than elsewhere. (The space cited here can be of either 1, 2, 3, ... or more dimensions). The minimal spanning tree technique is helpfulin cluster identification, thanks to its shortest link characteristic.
  In the computer world, a cluster also means a group of cpu's connected together.

- **Coherent** Light is coherent when it keeps the same frequency and the same phase during the emission.

- **Configuration** The configuration in the computer world means the way the computer components are disposed in a room as well as their inter-relations and quantity.

- **CPU** A CPU is a central processing unit. It is in fact the computer itself, including an electronic processor and the main memory.

- **Digitalisation** To what concerns computer data representation, one must know that these devices can only handle numbers. This means that to represent data, a combination of two binary digit can only represent 4 different states i.e. (0,0), (0,1), (1,0), (1,1). Then, to represent letters and other useful printing symbols, a set of 8 binary digits (or bits) are used, allowing the existence of 256 different characters. For the sound, compact disks use a 14 or 16 bits map to translate both intensity and frequencies of a sound. This means that a sampling is made to catch the state of a real event (or analogous event). The same kind of sampling is made for the images. So, digitizing means taking a photography of a reality, simplifying it. The computer image of a real world is always a model of that world, with a given resolution. Other information will be given under "image" and " resolution".

- **Entity** In project functionnal analysis, an entity is something concrete or abstracted, belonging to the perceived reality upon which one wants to record information. An entity does only exist considering the existence of related individuals who consider it as a solid and independant stuff with regard to its neighborhood and environment. An entity can be characterized by attributes. Ex.: a pencil, an eraser and a pupil are different entities.

- **Expert system** An expert system represent a complex and powerful computer software to help in diagnosis or identification. I.e.: MYCIN is a system to help in identifying the blood diseases. An expert system comprises the following parts:

  - a knowledge base (relations between symptoms and diseases)
  - a fact base (the observed symptoms)
  - a rule base (the rules to properly relate symptoms and diseases)
  - an inference engine (the "engine" that is going to deduce the diseases from the symptoms thanks to the rules)

  - a user interface (to allow a natural language interaction between the user and the system)
  - a proof supplier (to provide the different intermediate steps that led to the conclusion)

- **Filtering** When a filter is applied on an image, it is in most of the cases to get rid of some parts of the spectrum, that is to lessen the intensity of the i.e. red part of the light. But filtering can also be applied to transform the pixel intensities in the image according to two-dimensionnal mathematical laws such as Laplacian, Gaussian ... Transforming the intensities by saying: "every intensity value lower than 100 becomes 0" can also be considered as a filtering.

- **Fringes** An interferometer always produces on its screen a sequence of bands alternatively dark and clear. These bands are called fringes.

- **Fringe shape** The fringe shape is an evaluation degree of the curvature of the fringes: they can be either straight or slightly curved or circular.

- **Hardware** In the computer world, hardware means all the electronic components of a computer system.

- **Image** An image, is typically something that can be perceived by one of our sense: the view. Thanks to our eyes, sensible to the light we can *see* things that are illuminated. Many tens of years ago, Daguerre invented the photography to project the reality on paper. Since then, the system has been improved, giving the cinema, the colors, the TV... But these ways of keeping a track of the real world discretize it, not only in time (1 image every 24th of a second for the cinema), but also in space, since a photographic film is composed of a certain amount of tiny points each characterized by a local intensity and a local color. This implies that the resolution is limited: it means that the image won't be able to be much magnified.
  When an image is stored into a computer, the same phenomenon appears: the image is divided in small points called **pixels**. The particularity of the images commonly handled by computer is that they are coded in black and white. To have a color one, you must display three different images, recorded with different filters. Their merging will make appear the colored image.

A computer image is commonly a square of 512x512 pixels, each coded in 8,16 or 32 bits representing a grey level intensity with more or less accuracy. Appendix B presents the structure of a computer image.

- **Interactive** A program, or process, is interactive when its proper execution requires the presence of a human to enter commands at pre-defined points.
- **Interference** (optics). Interferences are produced by two light beams coming from different sources, these beams being in phase. See introduction for theoritical explanations. Interferences are commonly produced by an interferometer.
- **Interferogram** An interferogram is the image that appear on the screen of an interferometer. It always contains interference fringes.
- **Interferometer** An interferometer is a high precision optical device especially used for surface flatness evaluation. This instrument is composed of a source of coherent light (commonly a laser) and of a beam splitter that divide it in two parts. The former is reflected by a reference surface whose flatness characteristics are well known while the latter goes to the test surface. Both beams come back on a screen and interfere, producing bands of light alternatively dark and clear. The bands (also called fringes) shape can help determine the characteristics of the tested surface.
- **Interferometry** Interferometry is a high precision optical measurement method to evaluate surface flatness. The instrument used for this purpose is called interferometer.
- **Laser** The laser is an electro-optical device that can produce a very thin, coherent and monochromatic light beam. In this work, laser is only considered as part of an interferometer.
- **Link** In our problem, a link will represent an edge existing between two points; meaning that both points logicaly belong to the same structure. A link can be characterized by its length and by the identification of the points that it relates.
- **Merge** Doing a merge operation in a set of trees means in the context of this work to link two different subtrees together, by adding one new edge. The result is that the set of trees is diminished of one unit.

- **Modem** A modem (modulator-demodulator) is an electronic device used to realize long distance computer communications. The computer binary signal is transformed into an analog one, sent on a common telephone line, and decomposed into a binary signal on the other end of the wire to be understood by the remote computer.

- **Monochromatic** (for light). Monochromatic means that this light is composed of only one wavelength. So, it will be by no means white but red, green, cyan or ...

- **Neighborhood** Here, the neighborhood has to be understood the region surrounding a given pixel or set of pixels. The comparison of the intensities of these pixels with the surrounding region will allow the detection of maxima.

- **Noise** When one speaks about noise, it means undesirable sound, conflicting with what one really wants to hear. If the noise is too strong, the message or music cannot be understood. In image processing, "noise" is also used to mean unuseful and more or less equally distributed light in the image, that can, in the worst cases, hide the information.
  For the images, noise can be generated by the electronic components used to record the light; by the sky background in astronomical photography, by subinterferences in the case of interferograms. To give an example, if an astronomer is trying to count all the stars that he can see on a photography, he will have doubts while facing stars so faint that they can hardly be distinguished from the noisy sky background.

- **Operating system** The first computer program, unavoidable for the use of a computer is called operating system. It ensures the link between the hardware and the user's application programs.

- **Optical measurement** Optical measurement or **optical metrology** is the way to get information about the subject submitted to a test. These information are obtained using optical devices. Example: telemetry uses light to measure the distance between the light source and the target.

- **Orthogonal Quasi-Zernike Eigenfunctions**

- **Pattern recognition** Pattern recognition is a rather new topic in computer science. It tries to recognize a given object in an image or

a given sequence of words in the human speech. Most recent developments in this field try to give this ability to the computer through sophisticated programs. Pattern recognition is a field of artificial intelligence.

- **Peak** In an image, a peak will be a pixel or a whole set of pixels whose intensities are rather higher than the intensities observed in their neighborhood. The aim of the maxima filtering operation will be to detect these pixels.

- **Processing** The term "processing" always means, in computer science, the execution of a process in order to transform an input data set into a output data set.

- **Proto-link** When trying to merge parts of fringe structure together, link propositions are created and submitted to the appreciation of the merge process. These propositions can be called proto-links.

- **Resolution** Resolution is a measure of the density of information contained in a unit of the representation of the reality. For example, the resolution of a common photography can be 3000 points (or pixels) per square cm. The existence of resolution involve an important consequence: to enhance the size of a photography in order to discover new tiny details is unuseful above a given value.

- **Software** In the computer world, software represent all the computer programs that need the hardware to be run.

- **Table** For MIDAS a table is an array structured data set composed of lines and columns each representing an attribute of the data that this table describe.
  Example: if we want to represent fringes, we can store the x and y coordinates of the points that define it as well as the fringe number associated to each of these points. Then each table entry will represent a different point.

- **Tree** and **MST** "A tree is a connected graph with no circuits and a spanning tree of a connected graph G is a tree in G which contains all nodes of G. A minimal spanning tree (MST) of graph G is a spanning tree whose edge weight is minimum among all spanning trees of G". This definition is coming from [[zahn71]].

With a mathematical formulation, one could write:

Let $G = \{a_1, ..., a_n\}$ and let us build a tree.

$$\forall a_i, a_j \in G(i, j = 1, ..., n)$$
$$\exists 1\ L,$$
$$\wr a_i\ L\ a_j$$

in such a way that

$$\forall a_i \in G\ \ d^+(a_i) \leq 3\ and\ d^-(a_i) \leq 1$$

where d^+ and d^- mean the half external and internal degree of a_i and where the weigth of the link L written (w_{ij}) is in this case the length between a_i and a_j.

In an MST a supplementary condition on L appears:

$$w_{ij} = \min_{j \in 1,..,n} (dist(a_i, a_j))\ \forall\ i\ (j \neq i)$$

- **Smoothing** The smoothing of an image is similar to the application of a Gaussian filter to it.
- **Split** Doing a split operation in a tree means in the context of this work to cut a link in that tree separating like this the tree into two subtrees.
- **Window** The notion of window represent a frame that can be set on the pixels of an image. It means that we will only analyse the pixels that appear in this frame. The window (or frame) can have any size. Ex.: a 5x5 window will contain 25 pixels.
- **Wavefront** A wavefront is a surface of constant phase in a light beam.
- **Workstation** A workstation is a computer facility especially designed to offer the highest ease and comfort to the user: response time, nice screen and keyboard as well as a mouse or any modern pointing device. Each workstation comprises local processor, memory and a link to other workstation or to a central facility.

Appendix F

Listings

Two different kind of listings are available :

1. The FORTRAN codes of the different programs;
2. The procedure programs associated with the programs listed before. These procedures are written in the MIDAS command language.

The following listings will take place in the next pages :

- MAXIMA.FOR and MAXIMA.PRG associated with the new MIDAS command FILTER/MAXIMA
- POSITION.FOR and POSITION.PRG associated with the new MIDAS command POSITION/TAB
- MST.FOR and MST.PRG associated with the new MIDAS command MST/TAB
- SPLITMERGE.FOR and SPLITMERGE.PRG associated with the new MIDAS command SPLITMERGE/TREE
- EDITTREE.FOR and EDITTREE.PRG associated with the new MIDAS command MODIFY/TREE
- NUMBERING.FOR and NUMBERING.PRG associated with the new MIDAS command NUMBERING/TREE
- WAVE.FOR and WAVE.PRG associated with the new MIDAS command ANALYSE/WAVE
- ANINTER.PRG associated with the new MIDAS command ANALYSE/INTERFEROGRAM

In order to avoid paper waste, all the listings mentionned above won't be added to this book, however they are available on simple request by the author.

Bibliography

References on Automatic Analysis of Interferograms

[[becker82]] F. Becker, G. Meier, H. Wegner : automatic evaluation of interferograms.
SPIE Vol. 359 Applications of Digital Image Processing IV, p. 386; (1982).

[[becker85]] F. Becker, Y. Yu : Digital fringe reduction techniques applied to the measurement of three dimensional transonic flow fields.
OPTICAL ENGINEERING, Vol. 24, No. 3, p. 429; (Ma-Ju 1985).

[[choudry83]] A. Choudry, H. Dekker, D. Enard : Automated interferometric evaluation of optical components at the European Southern Observatory. Proceedings of SPIE (1983).

[[cline82]] H. Cline, S. Holik, W. Lorensen : Computer-aided surface reconstruction of interference contour.
Applied Optics Vol.21, No. 24, p. 4481; (Dec. 1982).

[[dekker85]] H.Dekker : Interferogram Analysis : some problems.
ESO Internal note (oct. 1985).

[[funnell81]] W. R. J. Funnell : Image Processing applied to the Interactive Analysis of Interferometric Fringes.
APPLIED OPTICS Vol. 20, No. 18; (Sept. 81).

[[haralick83]] Robert M. Haralick : Ridges and Valleys on Digital Images.
Academic Press. Inc. 0734-189X/83 Graphic & I.P.

[[nakadate81]] S. Nakadate, N. Magome, T. Honda, J. Tsujiuchi : Hibrid holographic interferometer for measuring three dimensionnal deformations.
Optical Engineering, Vol. 20,No. 2, p.246; (Ma-Ap. 1981).

[[nakadate83]] S. Nakadate, T. Yatagai, H. Saito : Computer-aided speckle pattern interferometry.
Applieds Optics, Vol. 22, No. 2, p. 237; (Jan. 1983).

[[pirenne85]] B. Pirenne, J. D. Ponz, H. Dekker : Automatic Analysis of Interferograms.
The Messenger (ESO) Dec. 85.

[[robinson83]] David W. Robinson : Automatic Fringe Analysis with a Computer Image Processing System.
APPLIED OPTICS Vol. 22, No. 14; (July 83).

[[trolinger85]] J. D. Trolinger : Automated data reduction in holographic interferometry.
Optical Engineering 24(5) Sep/Oct. 85 p. 840.

[[wyant82]] James C. Wyant : Interferometric Optical Metrology: Basic Principles and New Systems.
Laser Focus (May 82).

[[yatagay82/1]] T. Yatagai, S. Nakadate, M. Idesawa, H. Saito : Automatic Fringe Analysis using Digital Image Processing Techniques.
OPTICAL ENGINEERING Vol. 21, No. 3; (May/June 82).

[[yatagay82/2]] T. Yatagay, M. Idesawa, Y. Yamashi, M. Suzuki : Interactive fringe Analysis System : application to moire contourogram and interferogram.
OPTICAL ENGINEERING Vol. 21, No. 5; (Sept./Oct. 82).

[[zygo80]] Zygo Corporation : Interferogram Interpretation and Evaluation Handbook.
Ed.: Zygo Corporation. (1980).

References on Methods for Image Processing and Graph Manipulations

[[adorf85]] MIDAS – The Munich Image Data Analysis System.
ST-ECF Newsletter N° 4 December 1985, p.8

[[pavlidis82]] T. Pavlidis : Algorithms for graphics and image processing.
Springer-Verlag, Computer Science Press Inc., 1982.

[[pirenne83]] B. Pirenne : Représentation tridimensionnelle d'image digitalisées.
Travail de fin d'étude réalisé en vue de l'obtention du diplôme de gradué en informatique. (1983)

[[pirenne86]] B. Pirenne : The applicability of a tree structure based representation to the classification of galaxies.
ESO april 1986.

[[pratt78]] K. Pratt : Digital Image Processing.
Wiley (1978)

[[rohlf72]] F. J. Rohlf : Hierarchical clustering using the minimum spanning tree.
The Computer Journal Vol. 16, No. 1, p. 93; (1972).

[[rohlf78]] F. J. Rohlf : a probabilistic minimum spanning tree algorithm.
Information processing letters, Vol.7, No. 1; (Jan. 1978).

[[tanimoto77]] S. Tanimoto, T. Pavlidis : The Editing of picture segmentations using local analysis of graphs.
Communications of the ACM, Vol. 20, No. 4, P. 223; (Apr. 1977).

[[zahn71]] Charles T. Zahn : Graph theoritical methods for detecting and describing gestalt clusters.
IEEE Transactions on computers, Vol. 20, No. 1; (Jan. 1971).

References on Methods in Computer Science

[[aho74]] Aho, Hopcroft, Ullman : The design and analysis of computer algorithms.
Addison-Wesley (and Bell Telephone Laboratories (1974)).

[[bodart83]] F. Bodart, Y. Pigneur : conception assistée des applications informatiques; 1. étude d'opportunité et analyse conceptuelle.
Masson & Presses Universitaires de Namur (1983).

[[knuth73]] Knuth : Fundamental algorithms.
Addison-Wesley second edition (1973).

[[van lamsweerde84]] A. van Lamsweerde : Méthodologie de conception de logiciel.
Facultés Universitaires de Namur (1984)

References on the Physics of Interferometry

[[alonso67]] Alonso, Finn : Physique générale : Tome II, Champs et Ondes.
InterEdition, Paris (1967)

[[hecht75]] E. Hecht : Optique : cours et problèmes.
Série Schaum McGraw-Hill Inc., Paris (1983)

Users Manuals

[[ipg85]] Image Processing Group, ESO operating manual n° 1 : MIDAS Users Guide".
ESO-Garching (1985)

[[latex]] L. Lamport : Latex Document Preparation System.
Second Preliminary Edition (1983)

[[the messenger]] The ESO Messenger.
ESO-Garching (Dec. 1985)

[[tpu/editor]] Guide to text processing on VAX/VMS.
Digital Equipment Corporation Maynard, Mass. (1984)

[[vaxprimer]] VAX/VMS Primer.
Digital Equipment Corporation Maynard, Mass. (1984)